Precision Timing and Measurement for Inference with Laser and Vision

Alastair Harrison

St. Edmund Hall

Department of Engineering Science

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Michaelmas 2010

Alastair Harrison
St. Edmund Hall

Doctor of Philosophy
Michaelmas 2010

Precision Timing and Measurement for Inference with Laser and Vision

Alastair Harrison, St. Edmund Hall

University of Oxford

This thesis is about precise acquisition and inference of 3D range data. The intended application domain is mobile robotics. We investigate a number of key issues in the data gathering process, showing that a careful treatment of each stage in the pipeline is vital for producing dense accurate representations. We describe a low cost 3D laser system capable of generating high quality data from a continuously moving platform. The hardware, data capture, calibration and processing techniques we have developed allow us to produce remarkably detailed point clouds. Our laser system's rapid scanning of the environment enables the correction and augmentation of the robot's odometry system, by tracking planar features in consecutive sweeps of the environment.

An essential part of the data acquisition pipeline is accurate timestamping of data from different sources. We describe a new and very efficient algorithm for the rapid on line synchronization of computer clocks distributed over a network. The algorithm, known as TICSync+, is capable of achieving performance measured in Parts Per Billion and deals naturally with common clock upset events.

We also contribute a method for fusing point clouds with camera images to produce dense and accurate range maps of much higher resolution than the input range data. We make use of structural similarities in range and intensity data. Our use of a 2nd-Order smoothness prior allows the method to infer surfaces of arbitrary slope, as well as to reconstruct curved surfaces where appropriate.

Statement of Originality

This thesis is submitted to the Department of Engineering Science, University of Oxford, in fulfilment of the requirements for the degree of Doctor of Philosophy. This thesis is entirely my own work, and except where otherwise stated, describes my own research.

Alastair Harrison, St. Edmund Hall

Funding

This work described in this thesis has been generously supported by:

- UK EPSRC

  CNA and Platform Grant EP/D037077/1

- Guidance Ltd.

- Office of Naval Reasearch

  Grants N000140710550 and N000140810337 under Behzad Kamgar-Parsi

- European Commission

  Grant agreement number FP7-231888-EUROPA

We apologize for the inconvenience.

Normal service will resume shortly.

Acknowledgements

To properly thank my supervisor Paul Newman for his support over the past few years would require quantities of paper far exceeding the generous page allowance of an Engineering Department thesis. He has constantly encouraged, coaxed, enthused and amused[1], even despite an ill-conceived episode involving a hacked powerpoint presentation. I have undoubtedly gotten a far better deal than I deserved and am thoroughly indebted.

It has been a complete honour to work with the other members of the Mobile Robotics Group who are all quite incapable of being uninteresting. In particular I would like to thank Ingmar Posner for his friendship, advice and guidance.

To my friends, I apologize for ignoring you all summer; it was nothing personal. And to my parents, thank you for not getting too annoyed when I constantly deflected enquiries about the state of my thesis. It's finished now, OK?

To Kirsty, your patience and support has meant everything[2]. I'm looking forward to rediscovering the world outside the office with you.

[1]Though his use of the word 'futtock' in non-nautical contexts might some day wear thin.
[2]The surprise cakes and chocolate bars were pretty good, too...

Contents

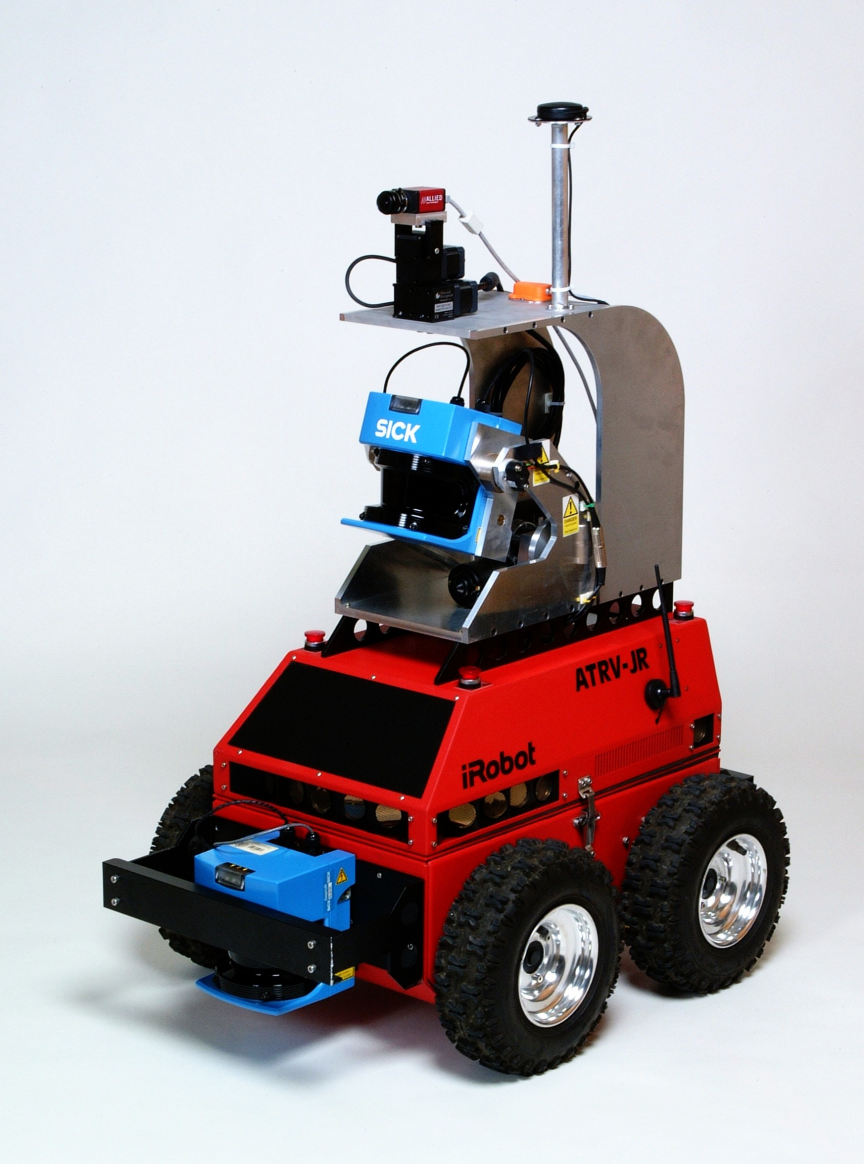


Figure 1: The robot vehicle we use to gather laser and camera data, known as *Marge*. A skid-steering system is used for directional control, with odometry obtained via encoders on the wheels. A camera with pan-tilt unit is mounted at the top. Immediately beneath that is the 3D laser scanner, which is the primary sensor discussed in this thesis. It covers a 70° elevation arc in around 0.6s, which allows it to rapidly acquire 3D range measurements of the robot's environment.

Chapter 1

Introduction

Workspace mapping and reconstruction by mobile robots is an important and growing field. Robots need to understand their environment and being able to reconstruct the 3D structure of their surroundings from raw sensor data is an important step in achieving this goal. In this thesis, a data processing pipeline driven by laser and camera data is proposed, with a view to producing and building quality 3D environment representations. Contributions are made in a number of areas, in particular those of calibration, point-cloud correction, laser and vision fusion and clock synchronization. This last point, clock synchronisation, is perhaps at first surprising particularly when one considers around 80 pages are given over to it. Robot perception is utterly dependent on the interpretation of time series data and the timestamps attached to data are no less important than the measurements themselves. Indeed we assert that the time-value tuple is an atomic entity. The value of a measurement is limited by the precision of its timestamp.

The contributions are put into context by first analyzing some existing work on 3D reconstruction from robot vehicles. It will be shown that the fidelity of reconstruction achievable is, as one might expect, directly related to the quality of 3D point cloud data used. Of vital importance to surface reconstruction techniques is a

way of obtaining good surface normal estimates. Noise in the point cloud data can significantly impact the normals.

This chapter will outline the structure of the thesis and provide a précis of our contributions to the topics covered. The reader is also directed to section 9.1 in the concluding chapter in which a perspective is given on the relationship between thesis structure and research chronology.

1.1 Point Cloud Collection

A low cost 3D laser measurement system is used, based on the well-known SICK LMS200 laser scanner. The LMS200 is placed in a nodding cradle (Figure 1) which briskly sweeps the laser's scanning plane over an elevation range of around 70°. The system enables us to gather dense and detailed point clouds, but demands that we pay very close attention to data timestamping issues. Even errors as small as 1 ms in the laser time stamps are significant, because it is vital to know the exact position of the nodding cradle when any given laser measurement is taken. Chapter 3 will describe the system in detail and show how very precise calibration is achieved using a novel algorithm that exploits the dynamics of the nodding cradle.

Point cloud data is often used in 3D *Simultaneous Localization and Mapping* (SLAM) systems, which aim to map unknown environments without recourse to infrastructure such as calibrated beacons or GPS sensors. The robot has only on-board sensors with which to determine its trajectory through the environment. One way of doing this involves using 3D point clouds to recover a full 6-DOF trajectory estimate. Point cloud segments are registered together using a technique such as ICP [11] and the transformation between them used to provide the SLAM engine with relative pose estimates. ICP requires that the two point clouds are internally well registered so it is common for the robot to follow a *stop-scan-go* paradigm for data gathering. A

less time-consuming method would be to allow the robot to move continuously while gathering data, but any odometry errors will cause the gathered point cloud segments to be corrupted.

Chapter 4 presents a method for correcting point clouds corrupted by poor odometry, by the use of architectural priors, such as the knowledge that urban environments often contain many vertical surfaces. By extracting and tracking planes in sparse atomic point cloud units, the trajectory of the vehicle can be recovered and the odometry corrected.

The next part explains the extrinsic calibration procedure for determining the transformation between the nodding laser scanner and a camera mounted on the same vehicle. Having the transformation allows laser measurements to be projected into an RGB camera image so that the pixels may be augmented with range information. From this, we can immediately produce coloured point clouds, with the colours coming from the RGB image, but we also have the information needed for probabilistically fusing laser and image data.

1.2 Image and Laser Fusion

One constraint on the speed at which a robot can gather point cloud data is the acquisition rate of the laser scanning unit. If the robot travels too fast, the point clouds will be too sparse to make high quality maps. On the other hand, high resolution camera images can be gathered at a much greater rate, though they lack range information. In Chapter 5 we will present a new method by which sparse laser data can be combined with high resolution camera images to produce an equally high resolution range image. The technique makes use of environmental priors such as that depth discontinuities often correspond to changes in colour, and that smooth surfaces tend to consist of smoothly varying colours. The contribution of a 2nd-order smoothness

term to a Markov Random Field (MRF) based representation allows the reproduction of high quality surfaces from remarkably little data. It will be shown that the algorithm compares favourably to other contemporary methods.

1.3 Clock Synchronization

The major contribution of this thesis is in the area of clock synchronization. Robotic systems are often distributed over multiple computers on a network, or even over the internet. High speed data gathering requires exceptional timing accuracy so that data streams from different sources may be fused correctly. Figure 1.1 shows another of the group's robots, which is highly dynamic and has very high bandwidth data gathering capabilities. A precise timestamping ability proved to be essential in the task of fusing the data from multiple sensors.

The clocks on standard PCs are notoriously inaccurate due to their low cost, temperature sensitive components. The traditional method of achieving clock synchronization is to have the participating computers synchronize to a Network Time Protocol (NTP) [70] server. NTP is an excellent, robust protocol, but it is designed for long term stability and robustness rather than rapid and absolute synchronization. In a robotics context where computers are often added or removed from a system at short notice, it is not acceptable to wait the few hours required for NTP to achieve synchronization.

Chapters 6, 7 and 8 introduce a new algorithm called TICSync, which uses two-way exchanges to synchronize clocks over a network. Rather than modifying the clocks, it learns the skew and offset parameters between them, so that knowledge of one clock's time will immediately allow recovery of the other's time. The thesis shows how the algorithm is able to naturally detect clock upsets and adjustments, such as those caused by running NTP.

TICSync is a very efficient algorithm, exhibiting constant time incremental updates and a small memory footprint. Performance is more than adequate, with ms accuracy being achieved after only a few seconds of message exchange. The performance is independent of the skew and offset between clocks. It depends only on the probabilistic network packet delay distribution. By modelling the distribution we are able to obtain good quantitative estimates of the accuracy of the algorithm at any point.

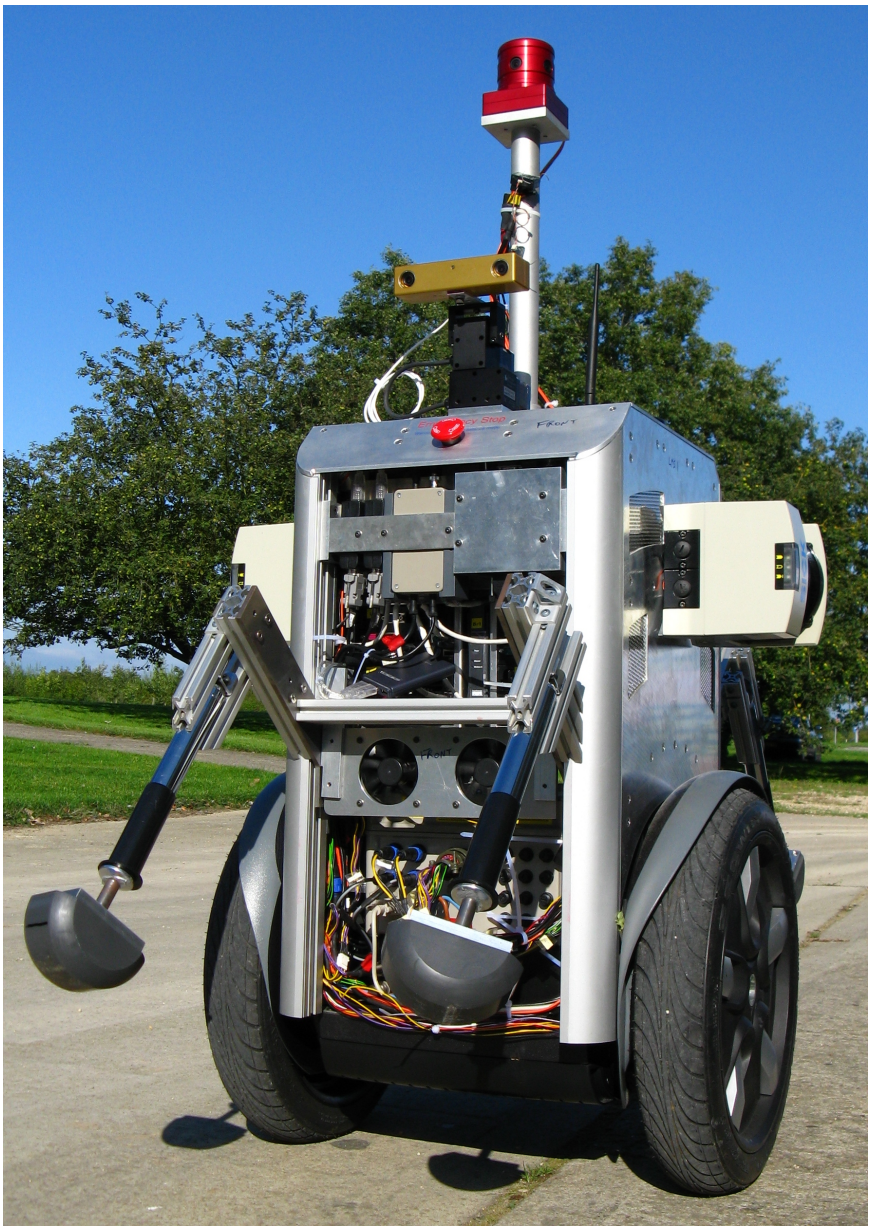


Figure 1.1: *Lisa* is a robot built for rapidly mapping large spaces. It was during the sensor integration phase of the build that the necessity of excellent time synchronization was discovered. *Lisa* is a balancing robot, constantly making many small adjustments to its pose, and therefore the sensors, which are rigidly attached. The sheer quantity of data being gathered was too much for one computer, so *Lisa* has four networked computers on board. Temporally matching the various sensor streams to produce consistent data proved intractable until the TICSync algorithm was implemented. The author's contribution in the construction of 'Lisa' was to design and build the power distribution infrastructure and to lead the hardware and software integration effort.

Papers arising from this thesis

[1] Cole, D., Harrison, A., and Newman, P. Using naturally salient regions for SLAM with 3D laser data. In *Workshop on Simultaneous Localization and Mapping, International Conference on Robotics and Automation* (Barcelona, Spain, April 2005).

[2] Dobnik, S., Pulman, S., Newman, P., and Harrison, A. Teaching a robot spatial expressions. In *Second ACL-SIGSEM* (Colchester, UK, April 2005).

[3] Harrison, A., and Newman, P. High quality 3D laser ranging under general vehicle motion. In *Proc. IEEE International Conference on Robotics and Automation (ICRA'08)* (Pasadena, California, April 2008).

[4] Harrison, A., and Newman, P. Image and sparse laser fusion for dense scene reconstruction. In *Proceedings of the 7th International Conference on Field and Service Robotics* (Cambridge, Massachusetts, July 2009).

[5] Newman, P., Chandran-Ramesh, M., Cole, D., Cummins, M., Harrison, A., Posner, I., and Schröter, D. Describing, navigating and recognising urban spaces - building an end-to-end SLAM system. In *Proc. of the Int. Symposium of Robotics Research (ISRR)* (Hiroshima,Japan, November 2007).

[6] Newman, P., Sibley, G., Smith, M., Cummins, M., Harrison, A., Mei, C., Posner, I., Shade, R., Schröter, D., Murphy, L., Churchill, W., Cole, D., and Reid, I. Navigating, recognising and describing urban spaces with vision and laser. *The International Journal of Robotics Research 28* (October 2009).

Chapter 2

Background

This chapter discusses previous work in the field of surface reconstruction from point clouds. A great deal of work has been done in the general area of surface reconstruction so only a flavour of the different approaches is given here. Some general techniques are described, as well as some which are specifically used for generating surface models from a mobile robot platform. We begin with a discussion of different environment representations and reconstruction methods.

2.1 Surface Reconstruction Methods

A non-mesh representation known as the *evidence grid* is described by Martin and Moravec [66], and is specifically intended for use by mobile robots. The world is represented by a uniform grid of *voxels*, with each one storing the probability of its occupancy. Whenever a new measurement of the environment is taken, a sensor model in the form of a probability density function (pdf) is used to update the occupancy probability of voxels in the vicinity of the measured target. As the robot moves around and takes more measurements, the quality and confidence of the map improves. The idea is elegant, but when applied in three dimensions the storage requirements are unacceptably large, especially when using a fine grid. Furthermore,

there is a requirement that the sensor's (and therefore the robot's) pose is accurately known when the measurement was taken, so that the correct voxels may be updated. In a general autonomous navigation setting, there is often uncertainty in the robot's current location

Many indoor environments consist of large, flat areas such as walls, floors, ceilings and doors. Detecting these planes in the point cloud and representing them with mesh elements can lead to significant storage reductions.

Nüchter, Surmann, Lingemann and Hertzberg [84] proceed by using RANSAC [38] to detect the plane with greatest support in the point cloud. Points supporting the plane are then projected in to the plane, and a quadtree technique is used to construct a mesh spanning them. This is necessary to reproduce the correct shape of the plane, including any holes it may have. The points are then removed from the cloud and the RANSAC algorithm re-run to find the next plane. This continues until no more planes of a suitable size are found. Triebel, Burgard and Dellaert [109] use the Expectation Maximization (EM) algorithm [30] to obtain planes more robustly and accurately, though it is a computationally expensive process.

Hähnel, Burgard and Thrun [45] propose another plane-extraction type algorithm for reconstructing both indoor and outdoor models. 3D point data is obtained using a 2D laser scanner, mounted pointing upwards such that it measures a complete vertical slice of the environment, perpendicular to the forward movement of the vehicle. Simple triangular meshing is performed by joining any three near neighbours, but these meshes are particularly noisy. To smooth the meshes, random seed points are chosen and a region growing method is used to find all neighbourhood points lying in the same plane. Neighbouring polygons in the region are then successively merged to produce a smooth plane.

Surface reconstruction from *range images* is a popular technique in the photogrammetry community. Using very high resolution 3D laser scanners that sample over a

regular grid, it is possible to generate a dense 2D image in which each pixel represents a depth measurement. This allows the application of standard image processing techniques such as smoothing filters. By applying edge detection algorithms, or by detecting other interesting features, a 2D Delaunay based triangulation can be obtained. Popping the points back to their 3D locations yields a feature sensitive surface mesh. Sequeira et al. [98, 99] use this for building indoor environment meshes. The results presented in those papers were significantly affected by noise, but by applying texture maps acquired from cameras, the eye can be fooled into believing that the mesh quality is better than it really is.

Range images are used to very good effect by Huber and Vandapel [53] to map underground mines. The high resolution laser scanner and relatively uncluttered surfaces in the mines contribute to the high quality models produced. Point clouds taken from a series of discrete poses are registered together and then the volumetric method of Curless and Levoy [29] (described below) is used to produce triangle mesh representations.

Schemes for reconstructing models of urban environments are often able to exploit some structure in the data, or other information available. Bauer et al. [9] take dense 3D scans of building facades and use RANSAC techniques to extract planes. The point clouds are further partitioned by using the knowledge that features such as windows tend to be axis-aligned with the facade, making them easier to detect and thus reconstruct and texture.

Früh and Zakhor [41] take vertical slice measurements of buildings as they drive past in a vehicle. Aerial images of the environment being mapped are then used to register the acquired point cloud correctly with the building fronts. From this, a pseudo range image can be constructed which may then be triangulated easily.

Some early surface reconstruction algorithms used deformation of an existing mesh to approximate the target surface. Miller and Breen [69] begin by embedding a

spherical mesh within the point cloud and gradually relaxing it to fill the space around it, until constrained by the point cloud. Where extra detail is required (determined from the error between the point cloud and the approximating surface) the mesh is subdivided and further relaxed. The method only works for closed surfaces which are known to be homeomorphic to the starting mesh, because there is no easy way to detect self intersections of the mesh. Consider inflating the mesh inside the ring of a doughnut; the two ends of the mesh can expand indefinitely if there is no way to detect when they meet.

Algorithms such as Cocone [1], Powercrust [2] and Eigencrust [58] create surfaces which are piecewise interpolations of all or some of the data points. The idea is to perform a Delaunay tetrahedralization of the point cloud (using [8] or similar) and to pick a subset of the faces to be part of the reconstructed surface. The techniques can be robust to outliers, but a noisy point cloud will result in a noisy surface.

Many algorithms work by first generating a signed distance function from the point cloud, then applying a standard isosurfacing technique to triangulate the zero set of the function, which interpolates the underlying surface. The method proposed by Hoppe [50] estimates surface normals by analyzing the local neighbourhood of each point in the cloud. The signed distance function at any query point is evaluated by measuring the distance to the nearest estimated surface tangent plane. The method is described in detail later.

Surfaces produced by the Hoppe method tend to be overly smoothed at sharp features. Hoppe et al. use a Mesh Optimization [51] technique to improve the surface fit to the point cloud. Using a combination of vertex perturbation, edge collapses, edge splits and edge swaps, the mesh is iteratively adjusted to reduce the distance between it and the point cloud. A useful side-effect is that the mesh is simplified in areas of low complexity.

The Hoppe technique is sensitive to incorrectly estimated normals, so Carr et

al. [17, 18] fit a smooth radial basis function (RBF) to the data, to diminish the effect. RBFs are expensive to compute, so an approximate algorithm called FastRBF is described, which speeds up evaluation and fitting. A similar technique is that of Ohtake et al. [85], which fits smooth functions to local regions of data, then blends them together in an Octree representation.

Volumetric methods are similar to the evidence grid method of Moravec. For example Pulli et al. [92] use a *space carving* approach to subdivide a volume into occupied and unoccupied voxels. Rays are traced from the sensor location, through a range image and into the voxel grid. Assuming the range image is adequately dense, if a voxel has a ray passing through it before hitting the surface, then the voxel is marked as empty. Curless and Levoy [29] use range images and a sensor model to build up a signed distance function for the surface, inside a voxel grid. They then apply standard isosurfacing techniques to extract the surface.

This literature survey has discussed key papers in the field. The next section will take what is considered to be the most promising approach (that of Hoppe [50]) and discuss it and possible extensions in detail. Our choice of the Hoppe's method is not central to this thesis, but it does serve to illustrate and provide an understanding of some of the issues involved in environment reconstruction.

2.2 Surface Reconstruction Overview

The surface reconstruction method of Hoppe, DeRose, Duchamp, McDonald and Stuetzle [50] is now described in detail. The method will henceforth be referred to as the Hoppe method. Many surface reconstruction techniques have been proposed since, but they are often intended for reconstructing small objects where noise is low and the surface sampling is always sufficiently dense. The Hoppe method is elegant and imposes few restrictions on the data, making it a good candidate as a basis for

a large-scale reconstruction algorithm. Some improvements are presented here, to make it more applicable to vehicle gathered data.

The Hoppe method consists of three stages, which are described here in detail. The algorithm takes as input a point cloud $X = \{\mathbf{x}_1, ..., \mathbf{x}_n\}$ and a set of viewpoints $V = \{\mathbf{v}_1, ..., \mathbf{v}_n\}$ where $X, V \subset \mathbb{R}^3$ and $\mathbf{v}_i$ is the location of the sensor when point $\mathbf{x}_i$ was sampled.

The technique begins by estimating a surface normal for every sample in X. These normals can then be used to generate a smooth signed distance function $f : \mathbb{R}^3 \mapsto \mathbb{R}$ which gives the shortest distance to the estimated surface from a given query point. The zero set $Z(f)$ is therefore an implicit representation of the sampled surface. Finally, the zero set is extracted by an isosurfacing algorithm such as Marching Cubes [62].

2.3 Tangent Plane Estimation

Each $\mathbf{x}_i \in X$ represents a sample of a surface, S in the environment being scanned. Even if the sample is noiseless, it tells us only that the surface passes through that point. By considering the location of nearby points known to be on the same surface, the local surface orientation can be estimated. The aim is to approximate S by a collection of flat tangent planes; one for each sample, $\mathbf{x}_i$. Each tangent plane $\mathrm{Tp}(\mathbf{x}_i)$ is parametrized by its centre $\mathbf{o}_i$ and a normal direction $\hat{\mathbf{n}}_i$. The planes are estimated using an orthogonal least-squares regression, which is described in Appendix A. Figure 2.1 shows an example point cloud, with computed eigenvectors and the fitted plane

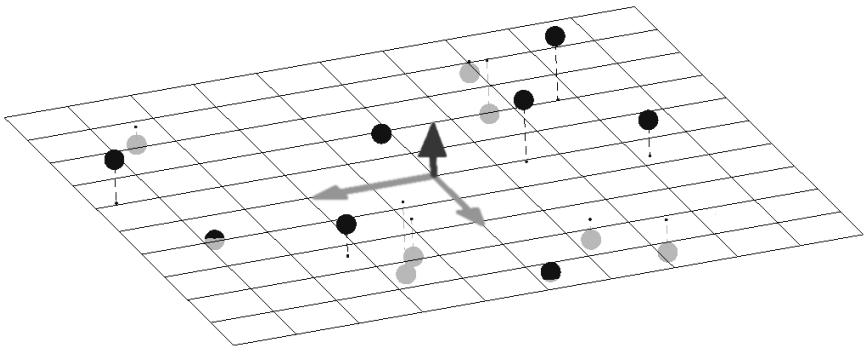


Figure 2.1: Fitting a plane to a set of points by minimizing the sum of squared distances to the plane. The eigenvector with the smallest eigenvalue (the vertical axis in this example) represents the plane's normal.

2.3.1 Initial Plane Fit

Consider a point $\mathbf{x}_i \subset X$ that lies on S, with the subset $N_i \subset X$ being the k nearest neighbours of $\mathbf{x}_i$. The assumption is made that the points in N_i also lie on S. It is also assumed that the surface patch occupied by N_i is locally flat. The consequences of these assumptions not being satisfied are discussed in Section 2.3.2.

Given the assumption of local surface flatness, the surface tangent at $\mathbf{x}_i$ may be estimated by performing a least squares plane fit to the neighbour set N_i. This will provide a point $\mathbf{o}_i$ which is the centroid of the neighbourhood and three orthogonal vectors $\{\hat{\mathbf{u}}_0, \hat{\mathbf{u}}_1, \hat{\mathbf{u}}_2\}$, with magnitudes λ_0, λ_1 and λ_2 respectively, such that $\lambda_0 \leq \lambda_1 \leq \lambda_2$.

The two longer vectors, $\hat{\mathbf{u}}_1, \hat{\mathbf{u}}_2$ lie inside the estimated plane, while the smallest vector $\hat{\mathbf{u}}_0$ is perpendicular to the plane and thus represents the plane normal, $\hat{\mathbf{n}}_i$. If all the points lie on the estimated plane then $\lambda_0 = 0$ and the fit is well conditioned. Pauly, Gross and Kobbelt [88] define a *surface variation* measure based on the relative magnitudes of the orthogonal vectors,

$$\sigma_k(\mathbf{x}_i) = \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2} \tag{2.1}$$

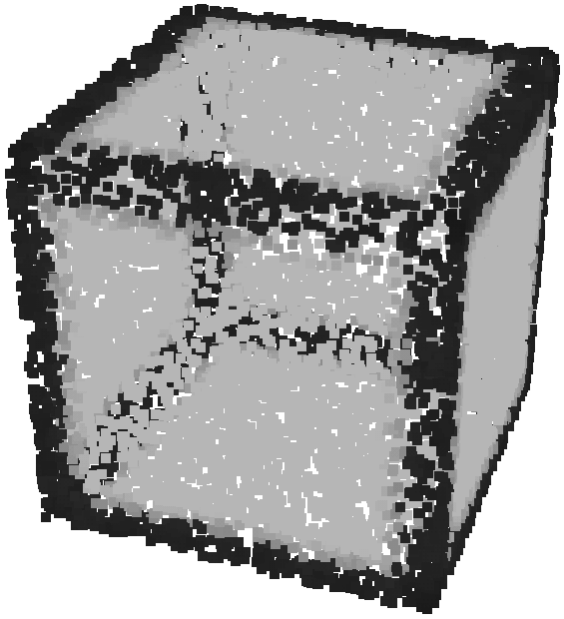


Figure 2.2: Variation of plane-fit conditioning across a point cloud sampled from a cube. Light areas are well conditioned and dark areas are badly conditioned. (plane fitting neighbourhood $k = 40$)

which gives an estimate of the deviation of the points from the fitted plane, or how well conditioned the fit is.

Finding the neighbour set N_i for all points can be a time consuming $O(n^2)$ operation if implemented naïvely. We use an efficient search structure known as the kd-tree [40] to reduce this to an $O(nk \log n)$ operation. Further details are in Appendix B.

2.3.2 Improving the Plane Fit

In areas of the point cloud affected by significant noise, the neighbourhood size[1] k may be insufficiently large to ensure a well conditioned plane fit. This is likely to be the case if the magnitude of the noise is similar to the radius of the local neighbourhood set N_i. It also happens if the point $\mathbf{x}_i$ lies near a sharp feature on the surface. This is demonstrated in Figure 2.2.

In the original work of Hoppe et al., this problem is mentioned but not explicitly

[1] Usually set to somewhere between 25 and 40

addressed; the sensor noise in their data was low, and the scanned objects were topologically simple. In processing data from a mobile robot, it is desirable to adaptively choose the size of the k-neighbourhood to reduce the influence of noise. If the surface variation measure is poor, then k is increased and the plane fit recalculated. This process is repeated until the surface variation falls below a threshold or k exceeds a predefined maximum. If at any step the surface variation measure increases then the poor conditioning is assumed to be caused by local surface curvature, rather than noise.

2.3.3 Consistent Normal Orientation

The surface S represents the outer boundary of solid objects in the environment, so it is helpful to think of S as having a *front* and a *back*. The *front* is the side seen from outside of the object and the *back* is the side seen from inside the object.

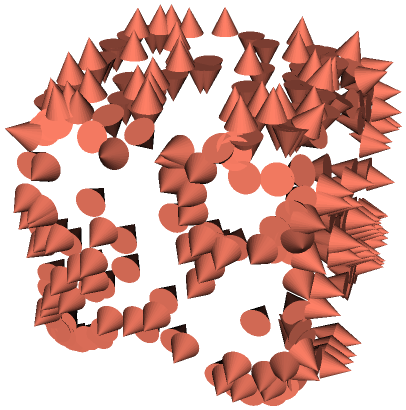
For illustrative purposes, Figure 2.3(a) shows a simple point cloud, sampled from three faces of a cube. Figure 2.3(b) shows the normals generated by the least-squares plane fit algorithm. Notice that they are all perpendicular to the surface, but are not consistently oriented. Consistent orientation is vital to the success of the reconstruction algorithm, so it is necessary to make sure that the normals all point out from the *same* side of the surface, whether that be the front or the back. This is achieved by *flipping* the incorrectly oriented normals.

Using Viewpoint Information

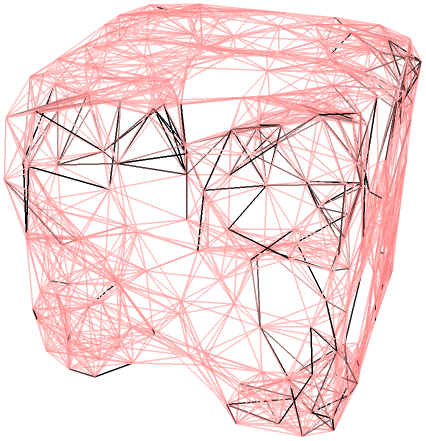
Given that the viewing location $\mathbf{v}_i$ is known for every sampled point $\mathbf{x}_i$, the obvious method of fixing incorrectly oriented normals would be to flip normals which point away from the viewpoint. The following inequality tests for incorrectly oriented normals:

$$\hat{\mathbf{n}}_i \cdot (\mathbf{v}_i - \mathbf{o}_i) < 0 \tag{2.2}$$

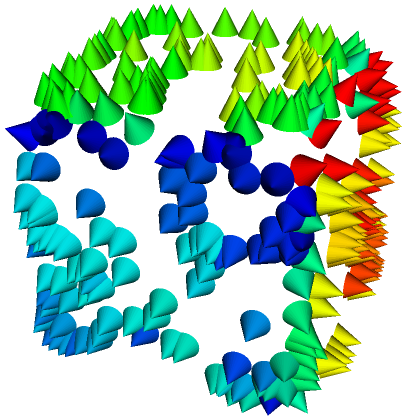
(a) Initial point cloud sampled from three faces of a cube



(b) Incorrectly oriented normals output from the plane fitting stage



(c) Neighbourhood graph edges defining order of propagation. Dark edges join vertices whose normals have greater disparity. Light edges join vertices with similar normals, indicating a smooth surface.



(d) Corrected normals after MST orientation propagation. The colours give an idea of the order of propagation; similar colours denoting similar times

Figure 2.3: An illustrative problem: finding consistent normals from a point cloud.

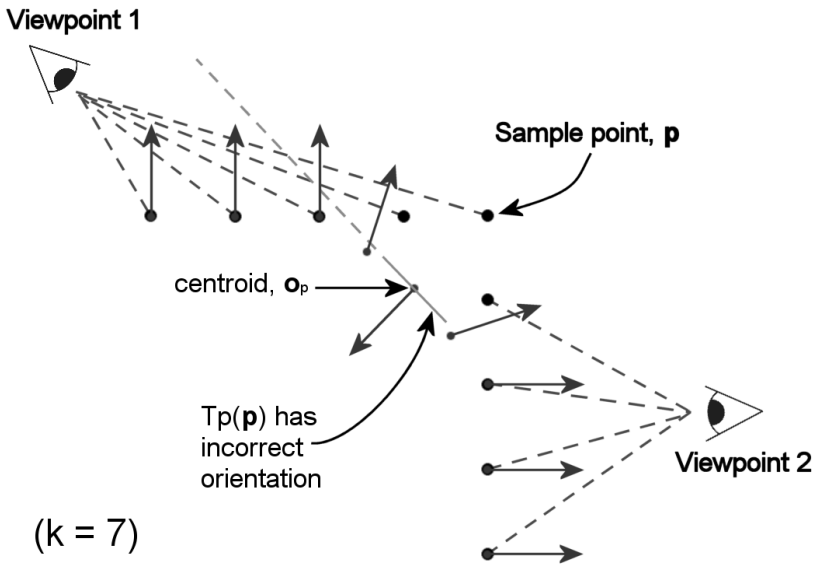


Figure 2.4: Using only viewpoint information to resolve normal orientation can result in errors. Point **p** is sampled from Viewpoint 1, at an oblique angle. Near sharp corners, the plane fit is ill conditioned, resulting in incorrect tangent plane estimates near sharp corners. Viewpoint 1 now lies behind the estimated tangent plane, causing the normal to be flipped in the wrong direction.

where $\mathbf{n}_i$ is the tangent plane's normal and $\mathbf{o}_i$ is the tangent plane's centre, obtained from the centroid of the neighbourhood set N_i.

While this works in most cases, there are some situations when the tangent plane is incorrectly estimated, causing some normals to be flipped unnecessarily (Figure 2.4). In general, tangent planes on surfaces perpendicular to the viewing direction are usually correctly orientated by the method. Where the surface is viewed obliquely, an ill-conditioned plane fit can result in the viewpoint being *behind* the estimated plane.

Hoppe proposed an interesting method based on Prim's algorithm [26] for generating minimum spanning trees. An extension to the method allows viewpoint information to be included where possible. A brief discussion of graphs and minimum spanning trees can be found in Appendix C.

MST Based Normal Orientation Propagation

Consider a pair of geometrically close points $\mathbf{x}_i, \mathbf{x}_j$ on a densely sampled surface, with estimated normals $\hat{\mathbf{n}}_i, \hat{\mathbf{n}}_j$. Assuming that $\mathbf{n}_i$ is correctly oriented to the front of the surface, $\mathbf{n}_i \cdot \mathbf{n}_j > 0$ implies that $\mathbf{n}_j$ is also correctly oriented. If $\mathbf{n}_i \cdot \mathbf{n}_j < 0$, then $\mathbf{n}_j$ is pointing from the *back* of the surface and must be flipped.

Once $\mathbf{n}_j$ is correctly oriented, neighbours of $\mathbf{x}_j$ can also be corrected. It is possible to continue propagating normal consistency in this manner until all points in X have been correctly oriented.

Hoppe et al. noticed that in practice, the order of propagation is very important. Propagation of normal orientations over sharp edges, for instance is prone to error and may result in inconsistencies. Further, the algorithm is greedy (decisions cannot be undone after they are made) so one incorrectly oriented normal may cause bad orientation to be propagated over large areas of the surface.

To improve the order of propagation, a graph $G = (X, E)$ is constructed. That is, the sampled points in X form the nodes of the graph and E is the set of (directed) edges in the graph. The graph contains an edge $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$ if $\mathbf{x}_j$ belongs to the k-neighbourhood of $\mathbf{x}_i$. Each edge is assigned a cost, $c_{ij} = 1 - |\hat{\mathbf{n}}_i \cdot \hat{\mathbf{n}}_j|$, so that $0 \leq c_{ij} \leq 1$. Therefore a pair of parallel normals would result in a cost of 0, but perpendicular normals would result in a cost of 1.

This *neighbourhood* graph, G is known as the Riemannian graph. The Riemannian graph for the point cloud from Figure 2.3a is shown in Figure 2.3c, where darker edges denote a higher edge cost (ie. higher curvature).

The correct orientation of the normals is propagated by traversing a *minimum spanning tree* of G, determined using Prim's algorithm. This method has the effect of propagating preferentially across areas of low curvature. Sharp edges are avoided unless there is no other path available to reach a particular point. Figure 2.3d shows the result of this process.

In the work of Hoppe et al., there was no viewpoint information available, so the propagation algorithm began by choosing a point at random and assuming that its normal was pointing out from the front of the surface. All the other points would then be made consistent with that.

Using Viewpoint Information to Improve Consistency

As discussed in section 2.3.3, using viewpoint information alone does not provide consistent orientation. It may however be incorporated into the Riemannian graph to improve results.

Given a set of viewpoints $V = \{\mathbf{v}_1, \ldots \mathbf{v}_m\}$, each viewpoint is added as a pseudo-node in G. Edges are added from each viewpoint node to every point 'seen' from that viewpoint. The edge cost is assigned as

$$c_{ij} = 1 - \left| \frac{\mathbf{v}_j - \mathbf{x}_i}{|\mathbf{v}_j - \mathbf{x}_i|} \cdot \hat{\mathbf{n}}_j \right| \tag{2.3}$$

Thus, where a surface is viewed obliquely the cost is high, ensuring that 'unreliable' viewpoint information is unlikely to be used.

2.4 The Signed Distance Function

Given a known surface, S and an arbitrary query point, $\mathbf{q} \in \mathbb{R}^3$, the signed distance $f(\mathbf{q})$ is, as defined in [50], "the distance between $\mathbf{q}$ and the closest point $\mathbf{z} \in S$, multiplied by ± 1". The value is positive when the query point lies in front of the surface, and negative when behind.

The signed distance can be shown to be an implicit representation of S, since $f(\mathbf{z}) = 0 \iff \mathbf{z} \in S$. Thus S forms the zero set $Z(f)$. Figure 2.5 shows a cross section of a surface, with associated signed distance function.

Using the tangent planes estimated in section 2.3, we can generate an approxima-

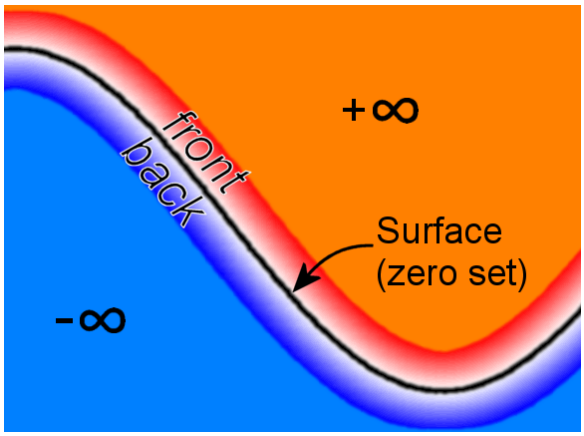


Figure 2.5: A cross section of a surface and its signed distance function. Red represents positive values and blue represents negative values. On the surface itself, the function always takes a zero value. Beyond a certain threshold distance, the function returns $+\infty$ in front of the surface and $-\infty$ behind the surface.

tion of the signed distance function of the surface, S, using an algorithm described by Hoppe et al. [50]. To evaluate the function at an arbitrary location, $\mathbf{q} \in \mathbb{R}^3$, their method first finds the estimated tangent plane $\mathrm{Tp}(\mathbf{x}_i)$ whose centre $\mathbf{o}_i$ is closest to $\mathbf{q}$. The distance is then calculated as

$$f(\mathbf{q}) = (\mathbf{q} - \mathbf{o}_i).\hat{\mathbf{n}}_i \tag{2.4}$$

which is the perpendicular distance to the tangent plane.

2.4.1 Isosurfacing

To visualize the implicit surface represented by a signed distance function requires a technique known as *isosurfacing*. An isosurface is the set of points of a given value in a scalar field - in our case we're interested in the *zero set* in particular. An isosurfacing algorithm extracts the surface into an explicit representation such as a polygonal mesh, which is easy to render.

There are a number of isosurfacing algorithms available [81]. The method due

to Bloomenthal [14] uses an adaptive subdivision scheme to converge on the implicit surface, and then replaces the subdivisions with polygon sections, which join up to form a mesh. In this work we have used a technique known as 'Marching Cubes', which subdivides the space into a set of small cubes, which are then replaced by different mesh sections, depending on the value of the implicit function at each vertex of the cube. The method is described in more detail in Appendix D.

2.5 ρ-density and δ-noise

Two quantities which are useful in detecting surface boundaries are ρ-density and δ-noise, described in [50].

We assume the noise model $\mathbf{x}_i = \mathbf{y}_i + \mathbf{e}_i$ where $\mathbf{y}_i \in S$ and $\mathbf{e}_i$ is the noise in the measurement. The point cloud can be described as δ-noisy if $\|\mathbf{e}_i\| \leq \delta \quad i = 1, \ldots, n$.

A similar measure can be derived for density. A set of noiseless samples, $Y = \mathbf{y}_1, \ldots, \mathbf{y}_n$ of the surface S are said to be ρ-dense if any sphere of radius ρ and centred on S contains at least one sample in Y. Thus, any sphere of radius ρ which contains no samples cannot intersect the surface S. The measure can be used to help decide whether a lack of samples in a particular area is due to a hole in the surface, or inadequate sensor coverage.

Because the signed distance function estimate considers only perpendicular distance to the nearest tangent plane, it can give misleading values at large lateral distances from surface boundaries. The effect is of continuing the zero set indefinitely beyond the boundary. If $\mathbf{z}$ is the projection of $\mathbf{p}$ onto $\mathrm{Tp}(\mathbf{x}_i)$, then using the definitions of ρ-density and δ-noise (see 2.5), Hoppe et al make sure that the signed distance is undefined when $\mathrm{dist}(\mathbf{z}, \mathbf{o}_i) \geq \rho + \delta$.

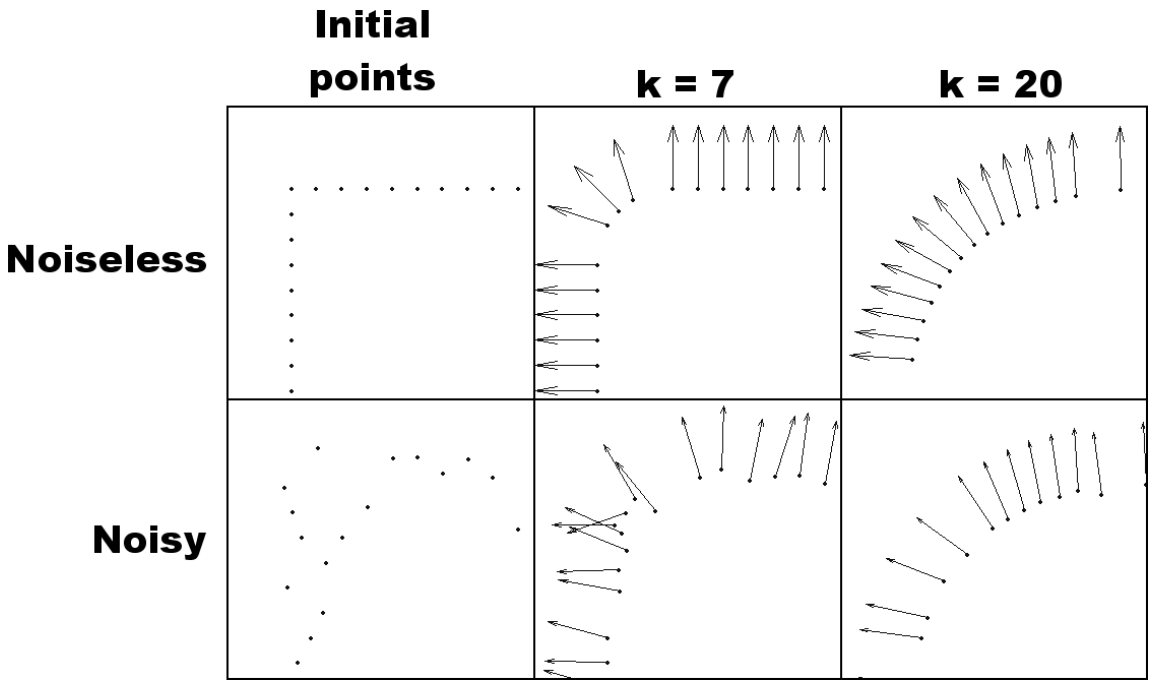


Figure 2.6: Fitting tangent planes to a neighbourhood of points acts like a Low Pass Filter over the surface. The first column shows a set of point samples near a corner feature, the lower set with noise added. The second and third columns show the effect of shifting each point to the cetroid of its local neighbourhood, for different neighbourhood sizes, k. With a sufficiently large neighbourhood size the noise is largely suppressed, at the cost of significantly smoothing the corner feature.

2.5.1 The Effect of Neighbourhood Size

The signed distance function described in Section 2.4 has a piecewise linear zero set. The surface extracted by the isosurfacing step can only be as smooth as the zero set it is derived from. Thus it is important to have the zero set as smooth as possible. Smoothness is controlled by the neighbourhood size used when fitting the tangent planes, and acts like a low-pass filter over the data. Larger neighbourhoods give greater smoothing, but also tend to round off sharp features, such as corners. There is also the increase in computation time to consider. Figure 2.6 demonstrates the effect of varying neighbourhood sizes on a simple 2D data set.

Using an adaptive growing approach can ensure that neighbourhoods are large enough to get a well conditioned plane fit. A disadvantage of this is that as the

neighbourhood size is increased, the centroid of the neighbourhood can move, along with the estimated tangent plane. This would occur particularly in noisy or curved areas. If the neighbourhood size changes rapidly across the surface, then the tangent plane centres may not line up smoothly, and the zero set can contain large discontinuities. The problem can be avoided to some extent by limiting the maximum neighbourhood size to a relatively low value. Points that seem to require a larger neighbourhood size should be removed or ignored if possible. As should points that are too far away from the centroid of their neighbourhood; this usually indicates an outlier.

2.5.2 Voronoi Regions and Pathological Cases

The signed distance function is estimated by finding the perpendicular distance to the query point's *nearest* tangent plane. This has the effect of limiting each tangent plane's influence to its Voronoi region (Figure 2.7). In the event that a poorly estimated tangent plane has a large Voronoi region, the zero set in that region will have large errors. Figure 2.7 shows the discontinuities well.

Insufficient Smoothing

Insufficiently large plane-fit neighbourhoods in a noisy area can have a disastrous effect on the distance function. Figure 2.8 shows an exaggerated example where the zero set is corrupted by noise.

Incorrect Normal Orientation

It should be apparent that consistency in normal orientation (Section 2.3.3) is vital. Figure 2.9 shows how one incorrectly oriented normal has the effect of flipping the distance function within the Voronoi region of the estimated tangent plane. The marching cubes algorithm relies on evaluating the signed distance function on a dis-

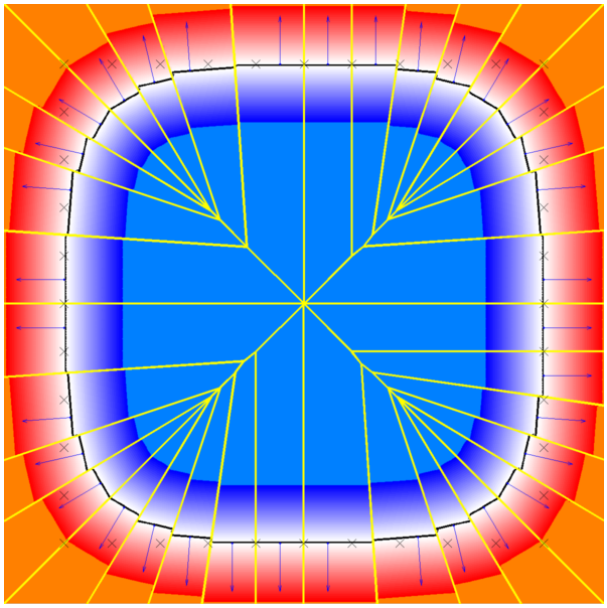


Figure 2.7: A cross section of an estimated signed distance function (SDF) for the original points shown as black crosses. Red denotes parts of the SDF deemed to be *outside* the surface and blue are those parts *inside* the surface. The blue blobs show the new centroids, computed from a neighbourhood size, $k = 8$. The arrows from each are the tangent plane normals. The zero set is shown as a black line which is piecewise linear through each centroid. The SDF is evaluated by measuring the perpendicular distance to the tangent plane closest to the query location. Thus, tangent planes only influence the function values inside their own Voronoi region (the boundaries of which are shown in yellow). The SDF has discontinuities at every Voronoi boundary.

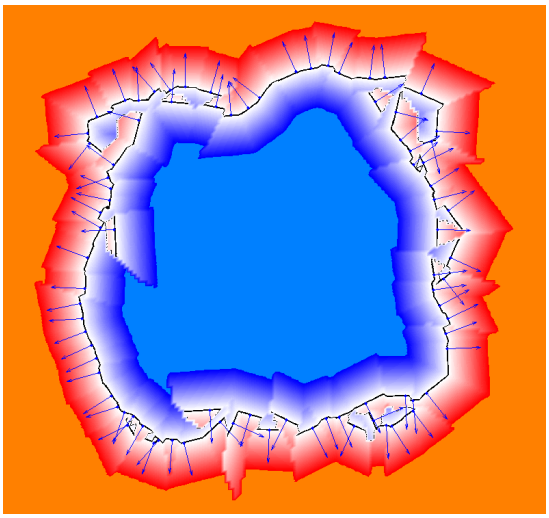


Figure 2.8: A cross section of an estimated SDF, with insufficiently large plane-fit neighbourhoods for the magnitude of the noise. There are many discontinuities in the SDF, so the zero set (in black) is also discontinuous and does not accurately represent the underlying surface which produced the point samples.

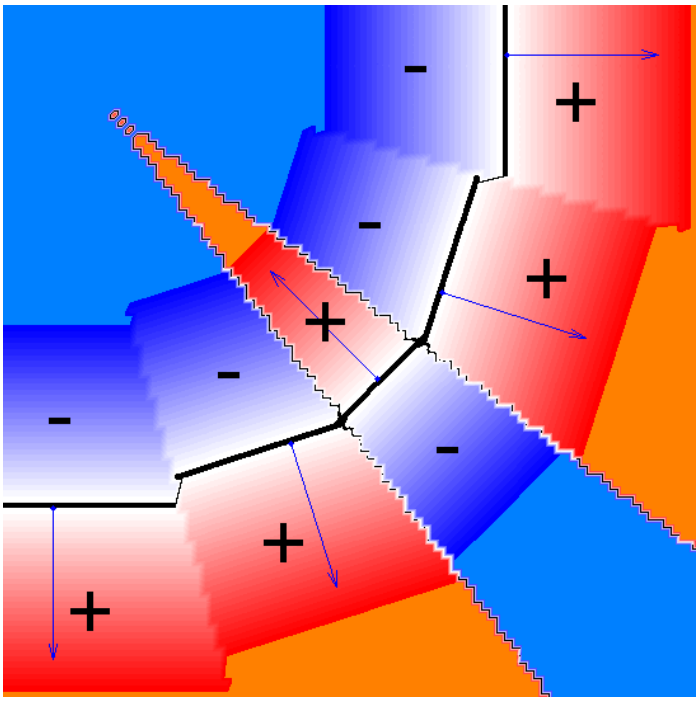


Figure 2.9: One normal has been *intentionally* flipped, to show the results of incorrect propagation. The thick black lines represent the piecewise linear tangent planes passing through each surface sample, with the surface normals shown as blue arrows. A given surface sample point can only affect the signed distance function within its own Voronoi region, as everywhere else is (by definition) closer to a different sample point. Flipping the normal of a given sample point causes the signed distance function to have its sign reversed in the local voronoi region. This results in a sign discontinuity along the boundaries of the voronoi region. The sign discontinuities are interpreted by an isosurfacing algorithm to be crossings of the zero set, thus producing spurious bits of surface in those areas. These are represented by the thinner black lines. Note that any jaggedness in the black lines in this image is caused by the sampling resolution of the signed distance function.

crete grid of points. If the function is found to be negative on one side of a voxel, and positive on the other, then the zero set is assumed to run through the middle of the voxel. Magnitudes are not taken into account. A similar process was used to generate the diagram of the signed distance function in Figure 2.9. The function was evaluated on discrete grid points and then filled in by interpolation. Notice how the zero set (represented by a black line) appears to separate from the surface and follow the Voronoi boundaries between two inconsistently oriented tangent planes. When marching cubes encounters a voxel which straddles the sign discontinuity in the SDF, it makes the incorrect assumption that the change in sign implies a zero crossing. If the Voronoi region is infinite, then the spurious surfaces at its boundaries will continue indefinitely. This is one reason for making the SDF undefined beyond a certain distance from the nearest tangent plane.

It is therefore vitally important to ensure consistency in normal orientation. The minimum spanning tree method (Section 2.3.3 works well, but occasionally makes a wrong decision. Its big disadvantage is that it is greedy and does not propagate any uncertainty measure when it makes an orientation decision. If one wrong decision is made, it is possible that it will be used as the basis for a series of later decisions, which will also be wrong.

As an example, consider the case of running the MST algorithm on a slightly noisy cube. The algorithm is correctly seeded at one point on the top face of the cube. Because the top face of the cube is relatively flat, all the tangent planes have almost parallel normals, so the correct orientation is readily spread. Once the entire top face has been covered, there are no further directions to go which do not involve propagating over a sharp corner. The algorithm will simply pick the least worst way to get over a corner. Suppose the orientation is not correctly preserved over the corner, because of noise. As soon as the corner has been negotiated, the algorithm encounters one of the smooth side faces of the cube, and will go on to spread the

incorrect normal orientation across the entire face.

The last example was somewhat contrived and the spread of these wrong decisions is fortunately limited by the use of viewpoint information where possible, but there can still be small 'outbreaks' of badly oriented normals, particularly in noisy regions. The point is that with the MST algorithm, the *best* decision is not necessarily the *correct* decision.

Insufficient Data

Figure 2.10 shows a typical outdoor laser scan and Figure 2.11 shows the estimated surface normals and the surface reconstruction. Some areas such as the trees simply do not have enough information to accurately reconstruct surface normals. As a result, the zero set in these areas resembles a honey-comb surrounding the Voronoi regions of the points. By making the signed distance function undefined beyond a threshold distance from the closest point, the honeycomb effect is limited to the local neighbourhood of the bad normals.

2.6 Chapter Summary

We have explored a number of existing environment reconstruction techniques and in particular the method of Hoppe [50]–a method that was chosen because it imposes few restrictions on the data. We have also touched on methods for efficient neighbourhood searches within point clouds.

Our investigation demonstrates the importance of high quality and consistently sampled point clouds for accurate surface reconstruction. The remaining chapters will treat each part of the data gathering pipeline with a view to maximizing the data quality.

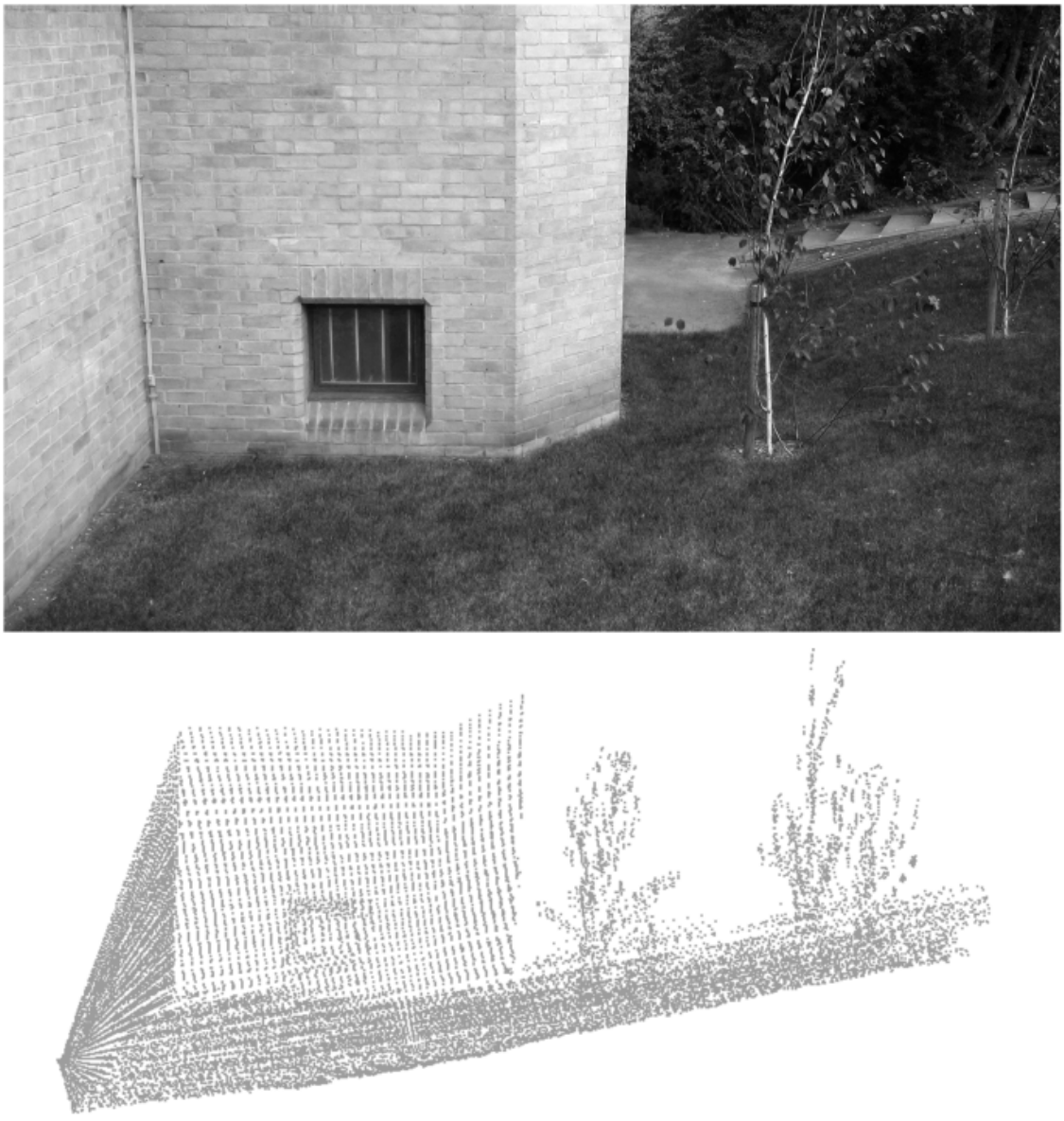
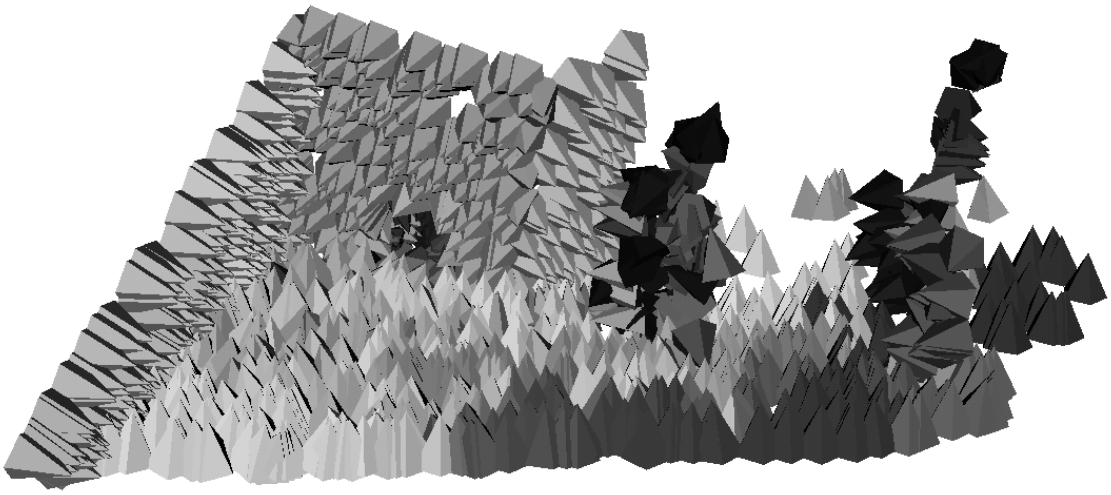
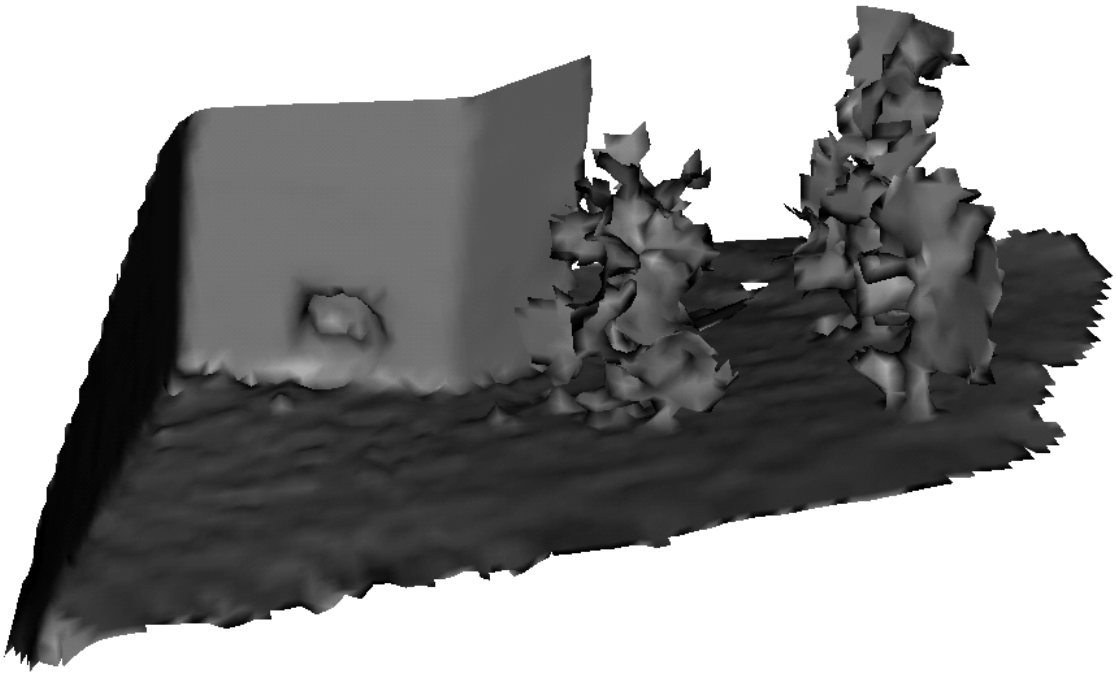


Figure 2.10: An outdoor location and the associated 3D laser data from a single pose.



(a) A subset of the calculated normals. Unsurprisingly, the normals in the trees are incorrect.



(b) The final marching cubes surface. Notice the honey-comb effect around the trees; neighbouring tangent planes have inconsistent normals, so spurious surfaces are created at the interface between their Voronoi regions.

Figure 2.11: Normals and final surface for the point cloud in Figure 2.10.

Chapter 3

Laser Hardware and Calibration

In this chapter we describe a 3D laser measurement system that is used to gather point cloud data from our robot. The system is based on a commercial off-the-shelf 2D laser measurement unit, with bespoke nodding apparatus. To obtain high-quality data, the unit must be well calibrated, and steps taken to ensure accurate timestamping of data. We present the algorithms and procedures developed to achieve the calibration. Later chapters will focus on methods to avoid relying on the bespoke hardware described here to address timing issues but this chapter serves to illustrate the importance of good timing.

3.1 Existing 3D measurement techniques

Laser measurement devices have become popular for use with mobile robotics platforms. They offer accurate range measurements of the vehicle's environment and high data rates. Though computer vision techniques such as stereo vision [97, 47] are increasing in popularity and performance, the 2D scanning laser range finder offers a higher level of robustness and is not dependent on surface texture and lighting to produce accurate results.

SICK LMS 2D Laser Measurement Systems are particularly favoured in robotics

due to their low cost, high accuracy and robustness. Once primarily used as a 2D sensor in mapping and sensing applications [63], they have more recently seen use in 3D data acquisition applications which sweep the scanning plane over the environment. Whilst custom optical solutions are possible [78], there are two more usual techniques for generating 3D measurements. First, the device may be fixed on the vehicle with a vertical scanning plane and the vehicle's motion exploited to move the scanning plane [104, 52, 106]. Alternatively the device may be coupled with an appropriate actuator which permits the scanning plane to be controlled independently of the vehicle's attitude [121, 105, 44, 71]. Use in this mode allows the collection of high-fidelity 3D measurements usually reserved for considerably more expensive devices. But the vehicle is required to be stationary during the sweep to ensure good coverage. We describe an actuated system capable of gathering accurate point clouds with the vehicle either stationary, or moving at a useful velocity, by virtue of its rapid sweeping.

For simplicity, the laser scanning system described here couples the SICK LMS200 with a low cost continuously nodding actuator rather than a servo. This does however present further timing challenges, since the nodding rate can be up to 180°/s.

3.1.1 The LMS200

Though capable of 0.25° resolution and 15mm accuracy, the LMS200 is designed primarily to be operated as a fixed sensor in industrial settings, so its use in mobile robotics can be problematic, particularly with regard to timing and data latency issues. This can have a significant impact on the quality of the data obtained from the device if not handled with care.

The SICK LMS200 operates by sending out infra-red laser pulses and measuring the time taken for the reflected light to return. The measured range is proportional to the time of flight of the pulse. Inside the LMS200, the laser beam is reflected by

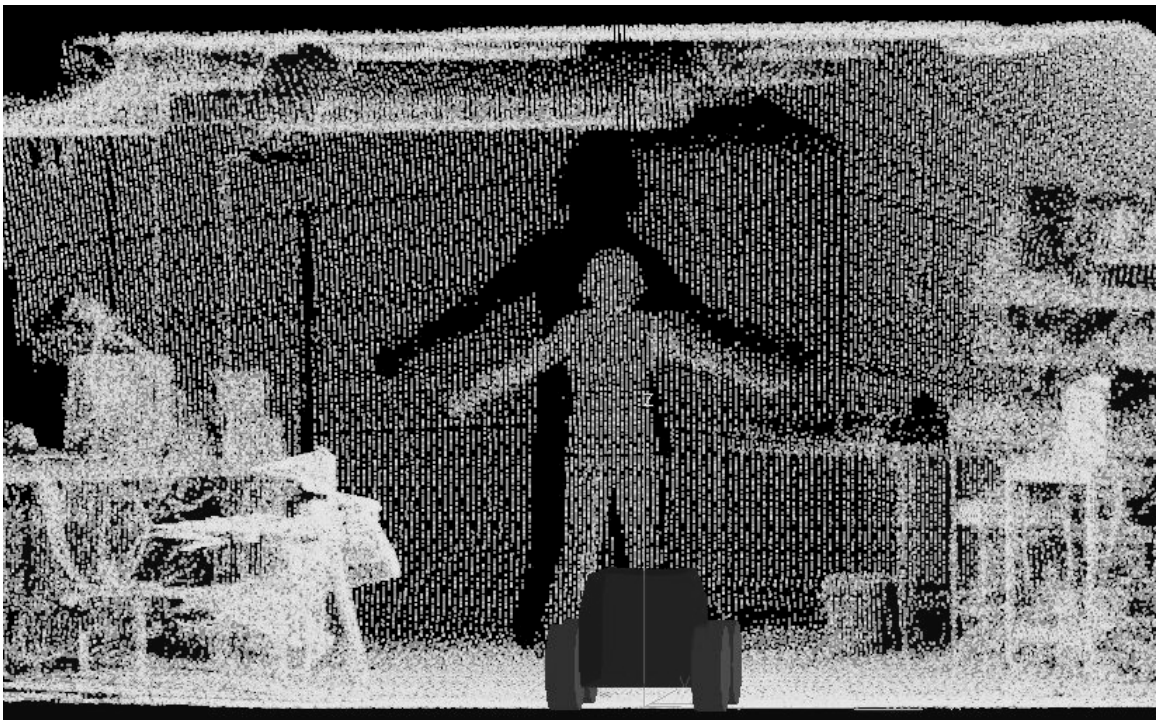


Figure 3.1: A typical point cloud generated from a stationary vehicle. The longer the scanning period, the more dense the data becomes, by progressively filling in the gaps.

a rotating mirror, allowing measurements to be taken over a 180° horizontal fan in front of the device. For the remaining 180° of the scan no useful measurements are taken because the mirror directs the beam inside the casing of the device. The mirror rotates at 75Hz and measurements are taken at 1° intervals, corresponding to a rate of 13575 measurements per second.

Usually, the LMS200 begins taking measurements when the mirror is in the 0° position, stopping at the 180° position. Optionally scans may be successively offset by 0°, 0.25°, 0.5° and 0.75°, to allow higher resolution coverage of the field of view. This mode therefore requires 4 scans to obtain full coverage. The LMS200 is capable of a number of other measurement modes, but the work described here uses the 180° scan/0.25° angular resolution mode, with a maximum range of 32m, at 1mm precision. When the vehicle is stationary this results in higher resolution point clouds, though there is little benefit when the vehicle is moving.

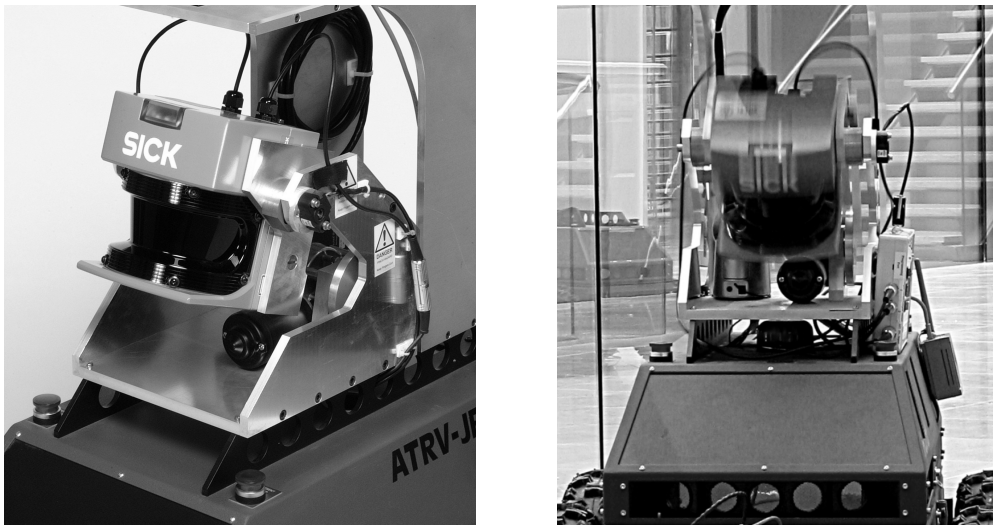


Figure 3.2: The nodding 3D laser range finder mounted on a mobile robot. In the middle of the elevation range, the laser moves at 180 °/s.

3.2 Nodding Laser System

On our robot, the LMS200 is mounted in a nodding cradle with a mechanical quick return mechanism and powered by a motor running from a fixed DC voltage source. The nod elevation profile is roughly sinusoidal, with a period of 1.2s. The elevation range is from +44° (upwards) to -26° (downwards), giving good, rapid coverage of the environment ahead. The maximum angular velocity occurs in the middle of the nod sweep and is around 180°/s. The apparatus is shown in Fig. 3.2. The nodder incorporates a rotary encoder for determination of the elevation angle.

The constant nodding scheme was chosen over a servoing scheme for simplicity, though it does place further demands on the data processing implementation, to ensure accurate timing.

3.2.1 Timing problems

The LMS200 communicates over an RS422 serial interface. While the mirror is in the front half of its rotation, measurements are gathered in an internal buffer. Once the

mirror reaches the back half of its rotation, the buffer contents are transmitted over the serial interface. If the mirror reaches the 0° position before the buffer contents have been fully transmitted then the following scan is discarded. In order for the device to transmit all scans, communications must be at the maximum rate of 500kBaud. This is a non-standard serial communications speed, requiring the use of either a USB-to-serial converter, or a special serial interface with a modified oscillator.

It is vital that we have an accurate time stamp on the laser measurements so that the nodder elevation angle may be determined with sufficient accuracy. The usual method of determining data timestamps from the time of arrival in the serial buffer is entirely insufficient for this task, since even a small timing error can produce gross vertical position errors with the maximum angular velocity of around 180°/s. The error may be estimated as $\zeta \approx r \sin(\tau \frac{d\gamma}{dt})$, where r is the measured range in metres, γ is the nod angle and τ is the timing error. Thus a modest 10 ms delay could cause an error of around 0.3m at 10m range. We shall describe a method of obtaining highly accurate correlation between laser data and nodder elevation angles.

The LMS200 has a capability to be synchronized with a second LMS200 in order that their mirrors are kept 180° out of phase, to eliminate laser interference [102]. The 'master' LMS200 outputs a 24V synchronization pulse whose falling edges coincide with the 0° position of the mirror. We use this pulse to trigger sampling of the nodder encoder, so that the precise elevation of the nodder is known at the beginning of each mirror sweep.

The system described here requires a computer with analogue data acquisition capabilities for sampling the nod angle encoder and a 500kBaud RS422 interface for communication with the LMS200. The precise set-up is not critical, but in this work we use a 1 GHz Kontron MOPS-lcd-VE based PC104 stack with 1GB RAM, with a Diamond-MM-AT data acquisition module and a CSM PCMCIA RS422 serial card (Figure 3.3). The operating system is a standard Linux installation; a real-time

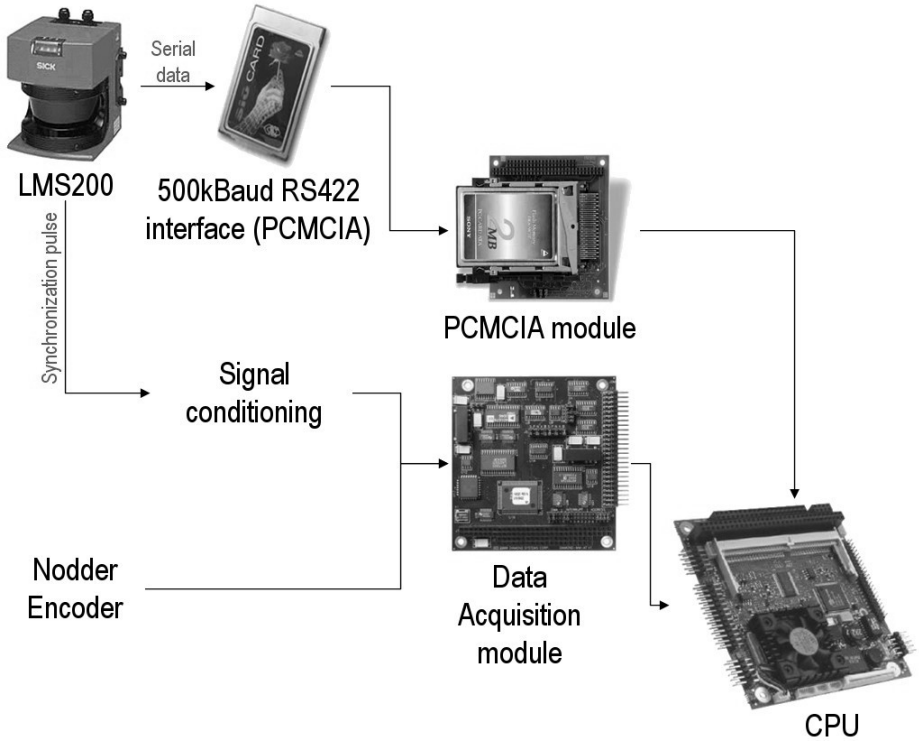


Figure 3.3: The data flow for data acquisition. The signal conditioning step takes the 24V synchronization pulse from the LMS200 and converts it into a 5V logic signal that can be used to trigger sampling on the data acquisition module

operating system is not required.

3.3 Calibration

Before use of the data, two calibrations must be performed – one for the mechanical system (a one time off-line step) and the other for the data acquisition system (an on-line step performed during start up). The software for data gathering, calibration and production of point clouds is implemented in C++ and runs in real time on the robot.

3.3.1 Mechanical calibration

The mechanical calibration is an interactive off-line step, which needs to be performed only once. It is necessary to accurately determine the minimum and maximum eleva-

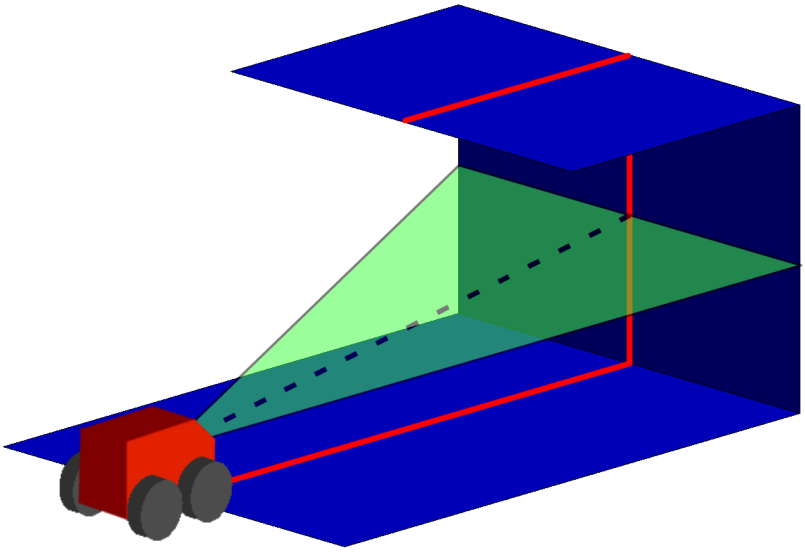


Figure 3.4: The robot is placed in an environment with a flat floor and a flat wall (not necessarily perpendicular to the floor). The middle range value from each scan is recorded, so that the calibration is performed on a vertical stripe through the environment.

tion angles $\gamma_{min}, \gamma_{max}$ of the nodder, so that they may be associated with minimum and maximum nodder encoder voltage readings. If the correspondence is inaccurate then vertical surfaces appear warped. Mechanical calibration of laser scanners is usually performed by scanning a known environment and then adjusting calibration parameters to minimize the error between the scan data and the environment model. Weingarten [117] uses a small room with very simple geometry that can be accurately measured. Underwood et al. [112] repeatedly drive past a known object and optimize the sensor pose to ensure the object looks the same on each pass. For our case, we are able to use a simpler environment.

We place the nodding apparatus upon a flat floor, facing a flat section of wall, though the wall does not need to be perpendicular to the floor. We define flat in this context to mean that the deviation due to curvature from a linear section is comparable to the noise of the sensor. The calibration considers only the middle range value of each laser scan; effectively a vertical slice in front of the sensor (see Figure 3.4). The first step is for the user to identify a set of points lying on a flat section of the wall, using a GUI. The task is then to optimize $\Delta = \gamma_{max} - \gamma_{min}$ such

that the wall points are as collinear as possible. Applying a common offset to both values simply rotates the measured points about the nodding axis, so the absolute values of γ_{min} and γ_{max} are not yet of interest.

The score for a given value of Δ is computed by performing a least squares line fit to the data points lying on the wall, and taking the sum of squared distances from the data to the line. Standard optimization techniques are then applied to find the minimum cost Δ - we used a hierarchical search. Finally, γ_{min} is chosen such that a straight line lying through the floor points is made horizontal. Figure 3.5 shows a few stages of the optimization.

3.3.2 Temporal Calibration

In order to convert laser scans to 3D Cartesian measurements, we must know the angle of the nodder when the scan was taken. We sample from the nodder encoder and laser at the same time (encoder sampling is driven by the laser synchronization pulse), so each laser scan has a single encoder measurement to which it should be paired. However the two streams of data arrive at the computer by two separate paths (laser by RS422 serial and encoder data through the ISA bus) and there is no identifying information by which they may be matched.

The obvious course of action is to use the timestamps attached to each measurement to match up the temporally closest pairs. Unfortunately this is not successful, because laser scans are taken at 13.33ms intervals and time stamping errors could be as great as 30-40ms. So for every laser scan, there is a small finite set of around 5 encoder samples to which it could feasibly be paired. Figure 3.6 shows how even a small timing error can cause significant corruption of the data.

A fact working in our favour is that both the encoder data and the laser scans always arrive in the correct order. If we can determine the appropriate allocation for just one pair of data points then we must also know the correct pairings for all future

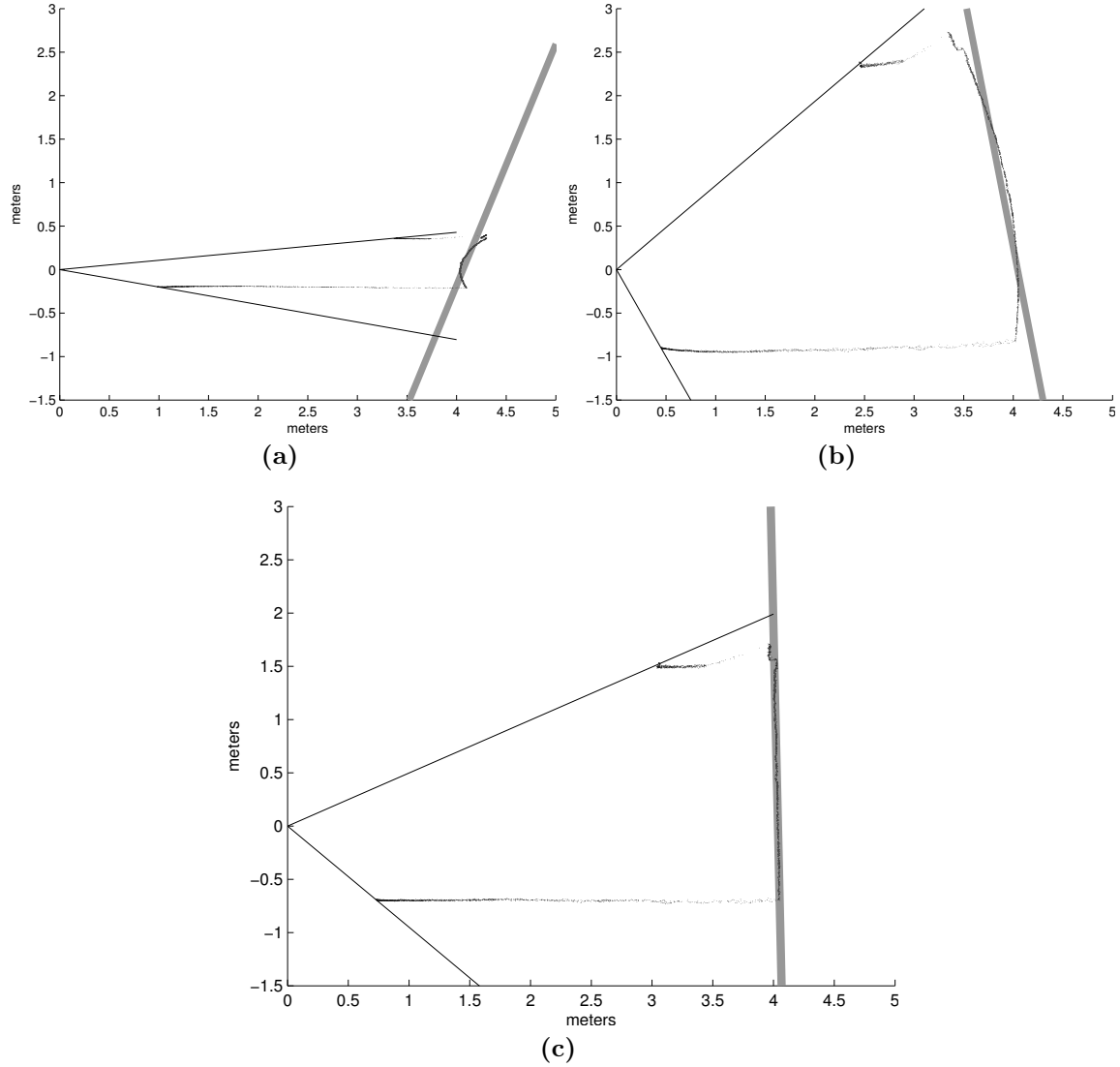


Figure 3.5: The calibration process to find the minimum and maximum nod angles. Data from a vertical stripe through the environment is plotted for three different ranges of minimum and maximum angles. The thick line is fitted through the points identified by the user as lying on the wall. The algorithm chooses the nod range which minimizes the sum of squared distances between the data and the line. Figure (c) shows the final result. In all three figures, the minimum nod angle was chosen such that the floor was horizontal, for visualization purposes.

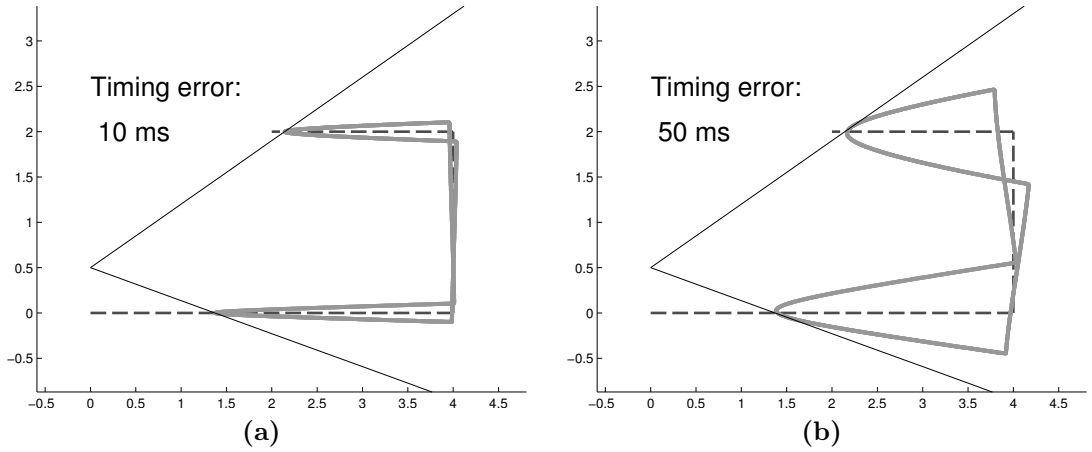


Figure 3.6: Synthetic data showing the effect of timing errors on a vertical slice of data produced by the nodding laser. Units are in m. The inclined lines are the nod limits, the dashed line is the correct surface (representing floor, wall and ceiling as in Figure 3.4) and the solid curve is the result for a given timing offset, for a complete cycle of the nodder unit. Correcting the timing error is equivalent to minimizing the area enclosed by the curve.

data. The problem is therefore one of finding the right alignment between a list of laser scans and a list of nodder elevation angles. Note that occasionally measurements can get lost in events such as buffer overflows, so it is important to keep track by taking note of scan IDs provided by the SICK laser. Using these, dropped messages are quickly detected.

This temporal calibration is performed in the sensor initialization phase and is designed to find the timing offset, τ between the two data streams. We begin by taking laser and nod angle measurements for a period T_{cal} (long enough to ensure a number of full nodder oscillations) with the vehicle stationary. We put no constraints on the environment, except that it must be static for the duration of the calibration. It is helpful to be in an environment with plenty of surfaces within the maximum sensing range of the laser so that there are enough measurements for the calibration.

As with the mechanical calibration, we consider only the mid points of the 2D laser scans; that is, a vertical slice directly in front of the sensor. If the y-axis is body-frame forward and the z-axis is body-frame up we have a time series of points lying in the Y-Z plane.

For a given trial offset $\hat{\tau}$, the measured points can be considered to be samples drawn from a continuous closed curve, traversed once for each full oscillation of the nodder. When $\hat{\tau} = \tau$ the curve will have zero enclosed area; the parts of the curve corresponding to upward and downward nod sweeps are coincident.

The curve may be parametrized in the Y-Z plane as $(y(t), z(t))$ where $0 < t < T_{cal}$ and its enclosed area given by

$$A = \left| \int_0^{T_{cal}} y(t) \cdot \frac{d}{dt} z(t) dt \right| \tag{3.1}$$

Since $y(t)$ and $z(t)$ are functions of range r and nod angle γ they may be rewritten as $y(r(t), \gamma(t))$ and $z(r(t), \gamma(t))$. Substituting these into (3.1) and noting that we wish to minimize the area, we find that

$$\hat{\tau} = \underset{\tau}{\operatorname{argmin}} \left| \int_0^{T_{cal}} y(r(t), \gamma(t+\tau)) \frac{d}{dt} z(r(t), \gamma(t+\tau)) dt \right| \tag{3.2}$$

In practice, the optimization to find $\hat{\tau}$ is straightforward. Assuming that timing errors are bounded, we are left with only a small discrete set of values of τ which we must test. Calculating the area enclosed by the curve can be achieved by approximating it as piecewise linear and joining up the sequence of measured points on the curve. Every line segment contributes the area between it and the z-axis. If the line segment is upward-going then the contribution is added, and if it is down-going then the contribution is subtracted from the total area. This method will only yield the correct area if the curve is closed: T_{cal} must be a multiple of the nod period, which is easy to arrange by detecting turning points in the encoder data. The steps are summarized in Algorithm 1.

Algorithm 1 Calculate the area of a piecewise linear curve

procedure CALCULATE-CURVE-AREA($\{(y_0, z_0) \,..\, (y_{N-1}, z_{N-1})\}$)
  $A \leftarrow 0$
  for $i = 1$ to $N - 1$ **do**
    $A \leftarrow A + (z_i - z_{i-1}) \left[\frac{y_i + y_{i-1}}{2}\right]$
  end for
  return $|A|$
end procedure

3.3.3 Transformations

Given an accurate elevation angle for every set of measurements received from the LMS200 it is an easy task to project the points into the vehicle frame. We do find it necessary to interpolate nod elevations across individual laser mirror sweeps, to account for the fact that it takes almost 7ms for the LMS200 to sweep its beam through 180°.

3.4 Chapter Summary

We have described our system for gathering high-quality 3D data from a mobile robot. We have shown that careful calibration procedures can greatly improve the quality of the data produced by the device. Two separate calibration methods were discussed, one to determine the mechanical limits of the nodding unit, and the other to recover the unknown timing offset between two streams of data. The latter made novel use of the dynamics of the sensor, resulting in an efficient algorithm that requires no special infrastructure or modifications to the environment. All it requires is that the environment remains static for the short duration of the calibration. The next chapters will consider how the data produced can be exploited.

Chapter 4

Data processing

Chapter 3 showed how accurate 3D data can be obtained from a rapidly nodding 2D laser scanner. With the robot stationary, the system is able to generate high quality point clouds of its local environment. For building accurate 3D point clouds over larger scales, the robot needs to move around to gather data, which requires an accurate localization ability. When suitable localization infrastructure (GPS, calibrated beacons etc...) is unavailable, *Simultaneous Localization and Mapping* (SLAM) [63] systems are required. SLAM provides a way for a robot to localize itself in an unknown environment using vehicle-mounted sensor measurements alone. We will describe a method of correcting odometry errors in point cloud segments so that they are suitable for use with scan-matching SLAM algorithms.

We will also show how to obtain a precise extrinsic calibration between the laser scanner and a camera also on the vehicle. This gives the ability to either augment the 3D point cloud with colour information from the camera images, or provide range information to pixels in the image. In the next chapter we will use the range-augmented camera image to produce high resolution range images of environments.

4.1 Point cloud correction

Nüchter et al. [83] employ an actuated nodding laser scanner to produce full 6-DOF SLAM maps of complex environments. They use a *stop-scan-start* paradigm to produce a series of intermediate point clouds, which are then registered against each other using the Iterative Closest Point (ICP) algorithm [11, 93, 39]. The relative transformations between the point clouds provide the vehicle pose constraints required by the SLAM algorithm. Having the robot stationary during scans means that the intermediate point clouds can be assumed to be well registered internally – a vital requirement for the point cloud registration step.

With the laser system described in the previous chapter, the rapid rate of nodding allows the environment in front of the vehicle to be scanned (albeit in relatively low detail) once every 0.6s. Measurements over larger ranges are noisier than those at short range. The faster the robot travels, the lower the sample density will be, which will limit the ability to reconstruct smaller features in the environment. We find experimentally that a speed of 0.5 m/s gives a good trade-off between data quality and coverage. A system capable of doing full 6-DOF SLAM on the data acquired by the moving robot is very attractive.

Cole and Newman [25] are able to produce 6-DOF SLAM maps using our nodding laser system with the robot moving continuously. The incoming data is divided into segments of a few seconds each, with the vehicle trajectory being recovered from wheel odometry. On a skid-steer vehicle, wheel odometry can in some cases be very poor - especially when changing direction on loose surfaces such as gravel. In order to ensure that each segment is internally well registered, the segmentation scheme used by Cole and Newman tries to ensure that a new segment is started whenever the robot changes direction (determined from wheel odometry readings). The scheme is successful provided that the robot's trajectory is carefully controlled.

Fig. 4.1 shows a point cloud gathered over a 10 second period, while the robot

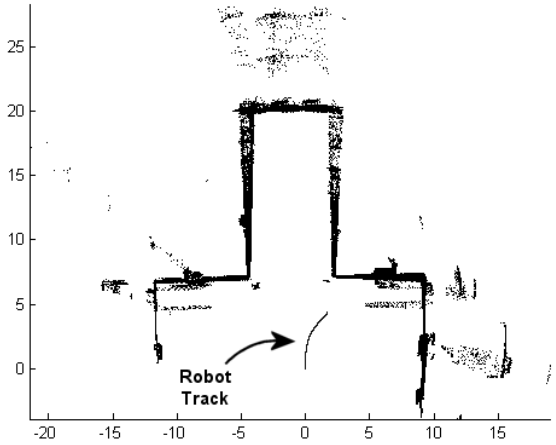


Figure 4.1: A plan view of an uncorrected 3D point cloud gathered while the robot was moving. The robot's track is shown in blue. Notice that odometry errors (wheel slip) cause gross alignment errors in the point cloud. Units are metres.

was moving without regard to producing good wheel odometry. There is significant noise in the data, particularly on surfaces seen from a distance. Even relatively small odometry errors can introduce gross misalignments over large ranges. Unmodelled pitching and rolling movements can cause similar data misalignments about their axes.

We now describe an algorithm which takes as input a point cloud with unknown odometry errors and produces a cleaned up point cloud, where rotations in all three principle axes have been corrected throughout the vehicle's trajectory[1]. This algorithm is implemented in Matlab and is an off-line step. Our prior is that most man-made environments contain many vertical surfaces. Exploiting the knowledge that almost vertical features in the point cloud almost certainly *are* vertical allows us to infer useful information about roll and pitch, where the data would normally be too sparse to perform operations such as point cloud registration.

[1]After this work was published, Bosse and Zlot [15] introduced a method of trajectory correction using a spinning 2D laser and continuous 3D scan matching. This was enabled by the development of a method for accurately registering sparse point clouds, despite the fact that surfaces are sampled at different locations in each cloud. They are able to recover highly dynamic trajectories from a vehicle moving at up to 6 m/s, with no wheel odometry, showing the power of the approach.

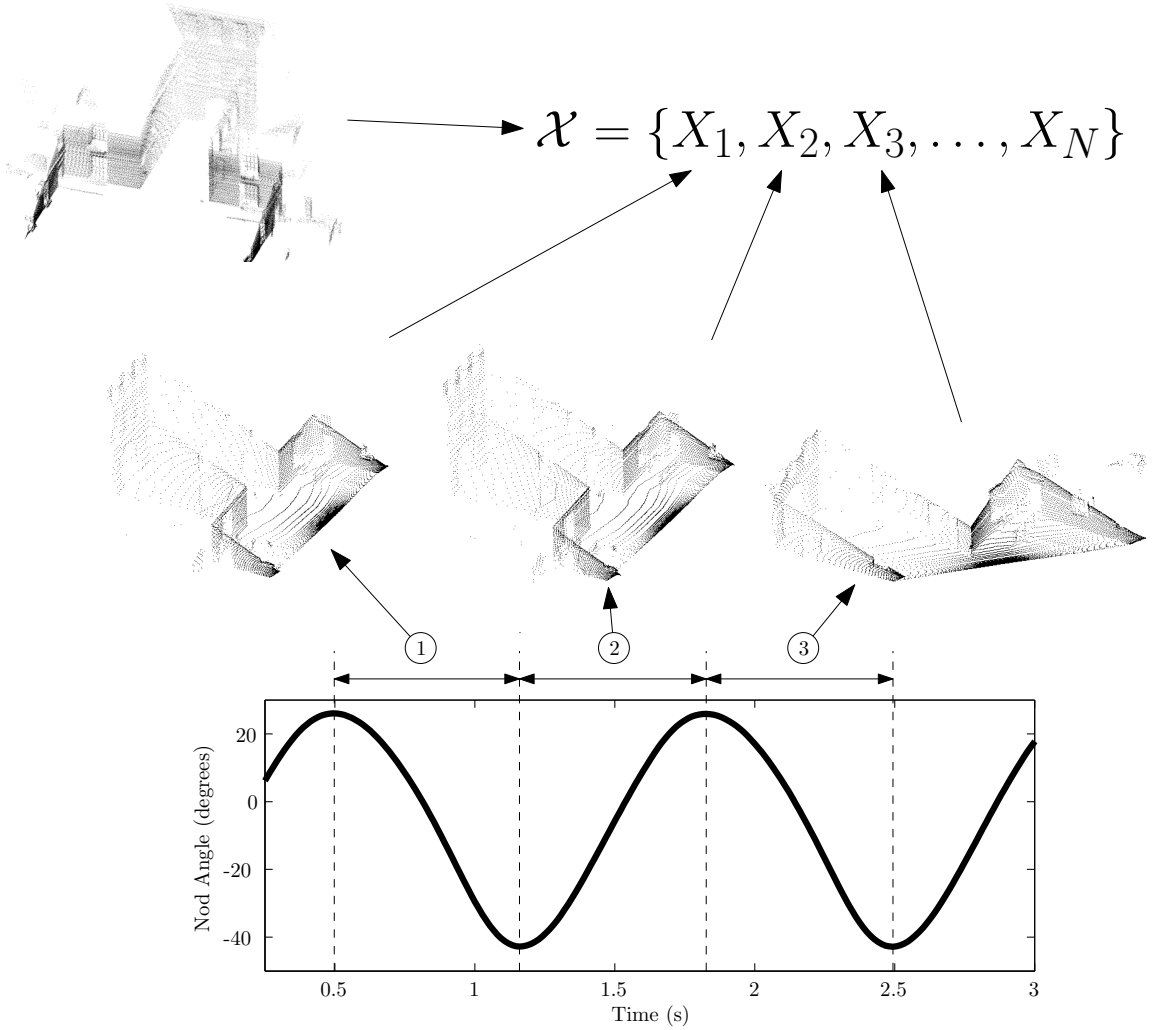


Figure 4.2: The point cloud $\mathcal{X}$ is subdivided into a set of smaller, temporally contiguous point clouds. These are produced from either one up-nod or one down-nod of the laser scanner. Note how sparse the points are in some areas.

4.1.1 Plane extraction

We start by splitting the point cloud, $\mathcal{X}$, gathered over a period T (usually around 10s), into a set of smaller, temporally contiguous point clouds:

$$\mathcal{X} = \{X_1, X_2, \ldots, X_N\} \tag{4.1}$$

where X_i is a set of 3D points collected over a suitable period, T_S. In our implementation, we choose T_S to be half the nodding period - the time taken for the nodder

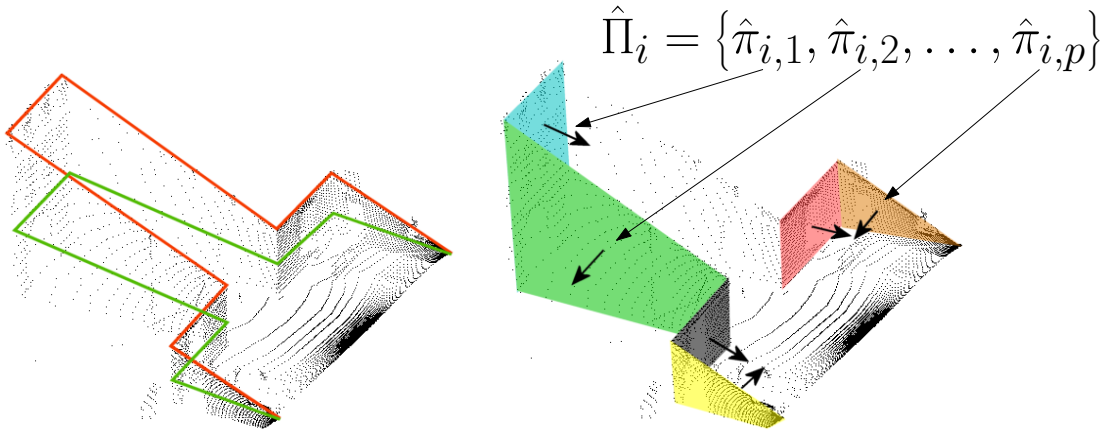


Figure 4.3: Lines are extracted from each laser sweep and are then clustered into planes.

to transit between its elevation extents. A few of the segmented clouds are shown in Figure 4.2.

From each cloud, X_i we first extract lines from 2D scans using a RANSAC based technique [79, 38], then cluster the lines over successive 2D scans to find a set of planes (Figure 4.3),

$$\hat{\Pi}_i = \{\hat{\pi}_{i,1}, \hat{\pi}_{i,2}, \ldots, \hat{\pi}_{i,p}\} \tag{4.2}$$

is the set of normals of the near-to-vertical planes extracted from X_i. For each X_i we seek a rotation matrix $R(\phi, \psi)$ where ϕ and ψ are rotations about the x and y axes such that

$$\hat{\pi}.\hat{\mathbf{z}} = 0 \qquad \forall \quad \hat{\pi} \in \hat{\Pi}_i \tag{4.3}$$

We wish to make all the near-vertical planes as vertical as possible by applying a single rotation to all of them. We do this by using the Matlab fminsearch function

to minimize the cost function

$$F(\hat{\Pi}_i) = \frac{\sum_{j=1}^{p} A_j \left[R(\phi,\psi)\hat{\pi}_{i,j}.\hat{\mathbf{z}}\right]^2}{\sum_{j=1}^{p} A_j} + w(\phi,\psi) \tag{4.4}$$

where A_j is the area of the plane whose normal is $\hat{\pi}_j$, such that

$$\phi_i, \psi_i = \underset{\phi,\psi}{\operatorname{argmin}} \left[F(\hat{\Pi}_i)\right] \tag{4.5}$$

The regularizer $w(\phi, \psi)$ penalizes deviations away from zero in ϕ and ψ when the ensemble of planes in $\hat{\Pi}_i$ inhibit robust estimation of ϕ or ψ. Consider a case where all planes have equal normals, all pointing along the x axis. Here the value of ϕ can have no effect on the value of F and may end up being set to an arbitrarily large value in the optimization. Thus we wish to prevent the optimization scheme from making unbounded modifications to ϕ or ψ. This motivates setting

$$w(\phi,\psi) = (1-\alpha)\phi^2 + \alpha\psi^2 \tag{4.6}$$

$$\alpha = \frac{2}{\pi} \arctan \frac{\sum_{j=1}^{p} A_j \left|\hat{\pi}_{i,j}.\hat{\mathbf{y}}\right|}{\sum_{j=1}^{p} A_j \left|\hat{\pi}_{i,j}.\hat{\mathbf{x}}\right|} \tag{4.7}$$

Figure 4.4 shows how the value of α varies with different plane configurations. With plane normals all facing the robot, roll is not recoverable and with plane normals all perpendicular to the robot's forward vector the pitch is not recoverable. The larger the plane, the greater the importance of its orientation. The area is hard to compute, so our implementation uses an approximation based on the quantity and average density of points supporting a plane.

4.1.2 Yaw slip estimation

We now introduce temporal groupings on planes, by searching for vertical planes which are similar through a sequence of contiguous point clouds, $X_k, X_{k+1}, \ldots X_{k+l}$ and

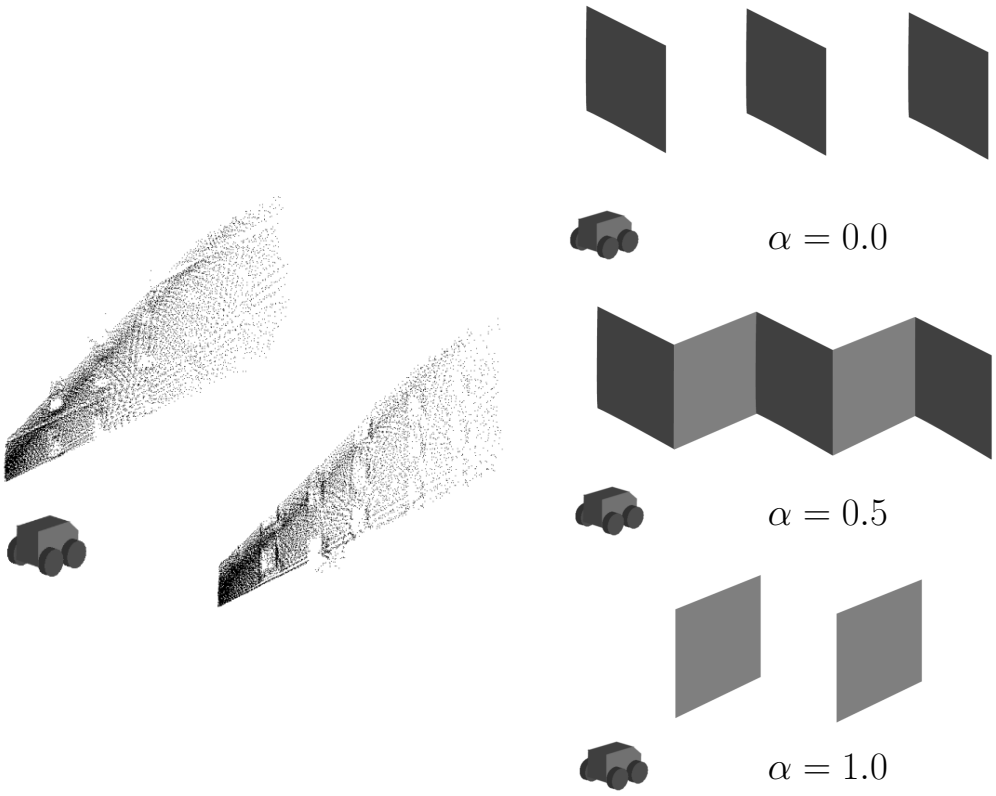


Figure 4.4: On the left is a case where only roll is recoverable; pitch is unconstrained. The right hand figure shows the values taken by the regularization term for various environment configurations.

assuming them to correspond to a single real-world planar surface. Plane similarity is determined by thresholding on angle between normals and distances to the origin for planes in consecutive clouds.

With zero slip about the yaw axis, these planes should be exactly coincident. In the presence of slip, the planes undergo rotational drift as shown in Figure 4.5. Given a pair of normals π_a, π_b representing the same plane at consecutive times, the instantaneous slip rate may be derived as

$$\omega = \frac{\arccos\left(\pi_a^T \pi_b\right)}{T_S} \tag{4.8}$$

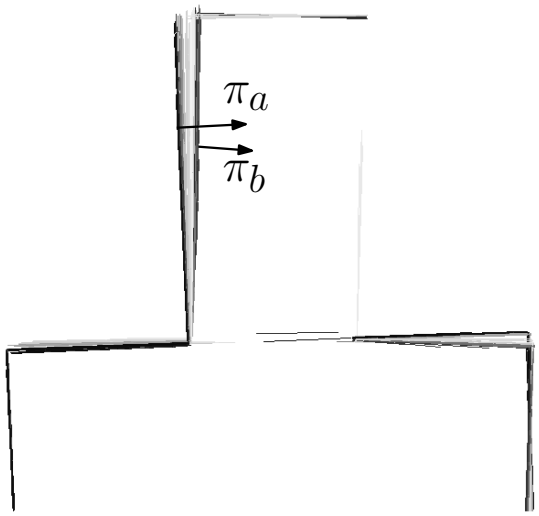


Figure 4.5: A plan view of grouped planes after vertical correction. The shading shows how they move over time, representing accumulated error in yaw.

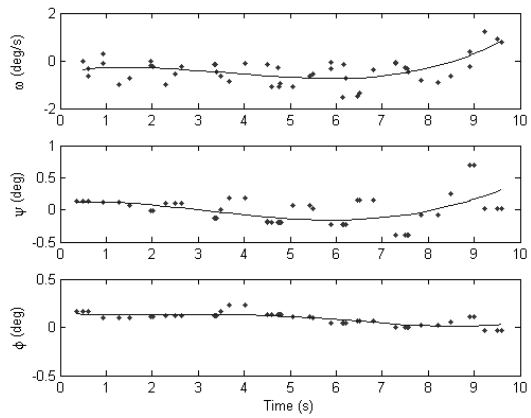


Figure 4.6: Yaw rate, pitch and roll adjustments throughout the time period of $\mathcal{X}$. The corrections are smoothed with polynomial fits of degree 4.

4.1.3 Reconstructing Point Clouds

At this point we have a discrete set of values of ϕ, ψ and ω for discrete points in time. By polynomial fitting we can obtain functions $\phi(t)$, $\psi(t)$ and $\omega(t)$ which are valid over the duration of the capture period T (Figure 4.6). Every individual 3D point x_i in $\mathcal{X}$ has a time stamp t_i and so in principal we can now retrospectively apply a correcting transformation $T(\phi(t), \psi(t), \omega(t), x(t))$. While the ϕ,ψ correction (which renders points in a gravity down frame) is simply a matter of pre-multiplication by a rotation matrix, $R(\phi(t_i), \psi(t_i))$, the ω (angular slip) correction requires more effort. For $0 < t < T$ we are in possession of the set of odometry measurements and thus interpolation allows us to express the odometric vehicle trajectory as a continuous function of time $\mathbf{x}_v(t)$. The corrected trajectory, $\hat{\mathbf{x}}_v(t)$ is deduced by integrating the yaw slip correction along the trajectory:

$$\hat{\mathbf{x}}_v(t) = \int_0^t \frac{d\mathbf{x}_v(\tau)}{d\tau} \oplus \begin{bmatrix} 0 \\ 0 \\ \omega(\tau) \end{bmatrix} d\tau \tag{4.9}$$

where $\oplus$ is the transformation composition operator. Now that we have corrected vehicle pitch and roll over the duration of the point cloud capture period, for each point x_i we can perform a final rotation around the yaw axis of the vehicle at time t_i to yield the final corrected point cloud $\mathcal{X}'$.

Figure 4.8 shows before-and-after views of the point cloud in Figure 4.1. Figure 4.7 shows a collection of corrected clouds rendered together as part of a SLAM map.

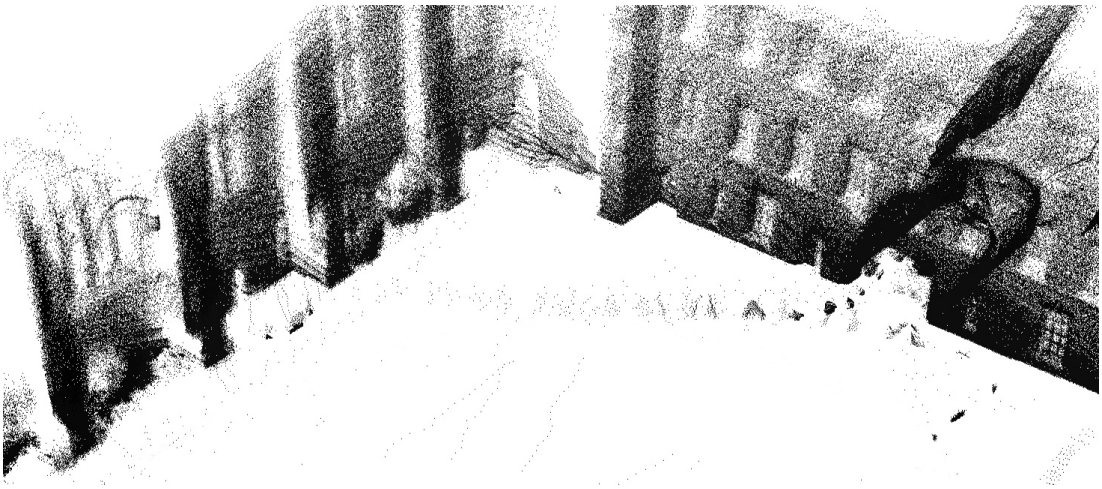


Figure 4.7: A collection of corrected point clouds fused together in a SLAM framework.

4.2 Extrinsic Calibration of Sensors

Now that we have access to more accurate laser data with the robot moving, we look at a way of adding value to the data by fusing information from a camera also located on board the robot. In order to fuse camera and laser data it is necessary to accurately locate both sensors in each other's frame of reference. This section describes the calibration process which we employ to find the transformation between the two frames.

Given a set of points $\mathbf{X_L} = \{x_1, x_2, \ldots, x_n\}$ in the nodding laser's coordinate frame, we wish to find a rotation $\mathbf{R} \in SO(3)$ and a translation $\mathbf{T} \in \mathbb{R}^3$ that transform the points into the camera's frame, so that:

$$\mathbf{X_C} = \mathbf{R}\mathbf{X_L} + \mathbf{T}, \tag{4.10}$$

where $\mathbf{X_C}$ are the points in the camera's frame.

We assume here the that camera's intrinsic calibration parameters are already known, having been calibrated with standard techniques such as those of Tsai [110] and Zhang [124]. We used the Matlab Camera Calibration toolbox [16] to calibrate our camera. Having the intrinsic calibration allows us to project the points $\mathbf{X_L}$ in to

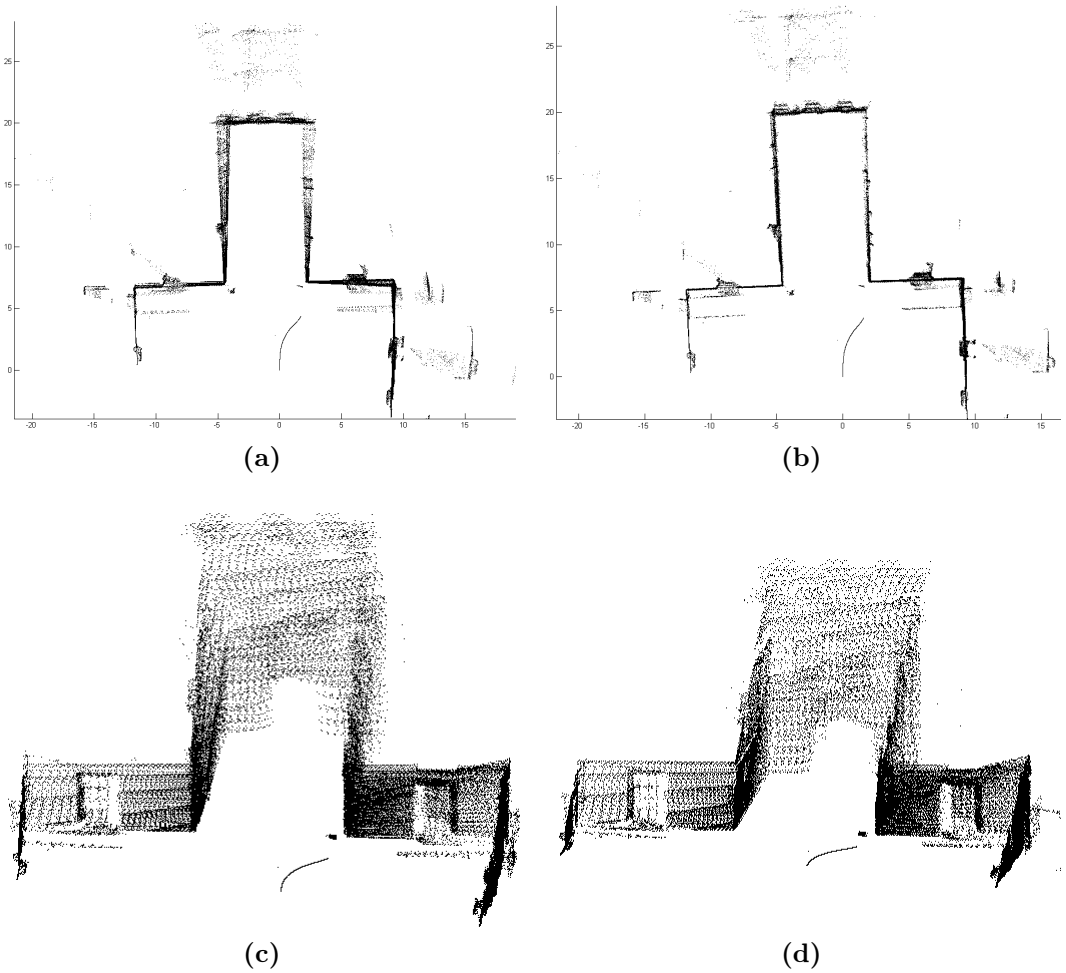


Figure 4.8: Figures 4.8a and 4.8c show a plan view and oblique view of an uncorrected point cloud, first seen in Figure 4.1. Figures 4.8b and 4.8d show the corrected versions. They are noticeably crisper, as the rotation errors have been greatly reduced.

the image, once we have first transformed them into the camera frame.

4.2.1 Existing Extrinsic Calibration methods

Unnikrishnan and Hebert [113] suggest a method involving a checkerboard style calibration target similar to that used in intrinsic camera calibration. The target is imaged by both sensors in a variety of orientations and then the user manually identifies the corners of the target in all of the pictures and laser scans. A least-squares procedure is then used to minimize the distance and orientation error between planes

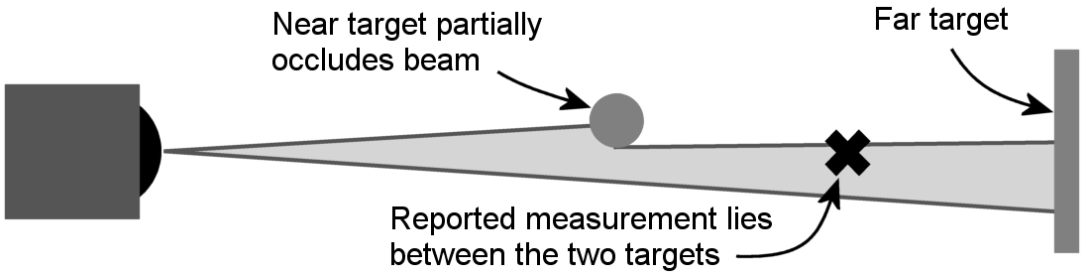
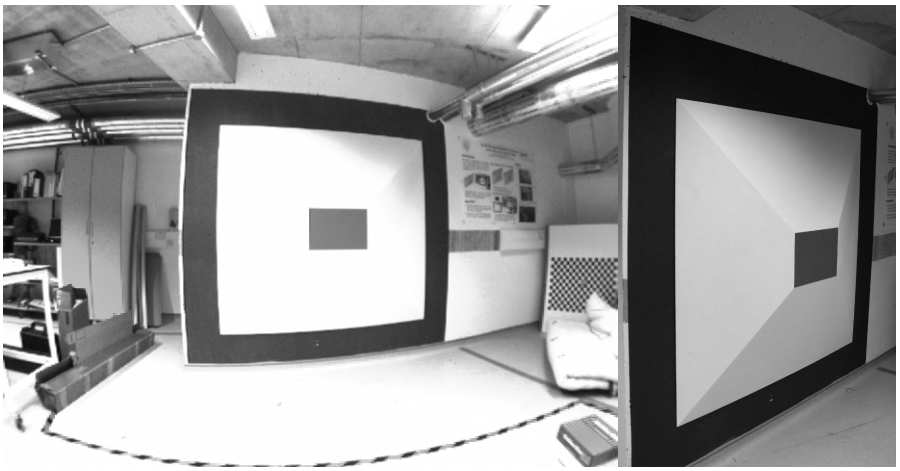


Figure 4.9: Mixed measurements occur when the laser beam is reflected from two targets at different depths. The reported measurement lies somewhere between the two targets' depths.

in the images and laser scans. Cobzas et al. [21] show how to perform an extrinsic calibration when the laser points are visible to the camera. Scaramuzza et al. [95] present a camera image and a range image to the user and have them manually select correspondences. They process the range image to enhance corners and edges to make this task easier. The user must provide enough correspondences so that errors introduced by the selection process can be averaged out. Zhao et al. [125] show how to find extrinsic calibration parameters for a number of sensors on a vehicle, but again they require manual selection of corresponding feature points from each sensor.

Posner et al. [90] employ a specially constructed calibration target which may be robustly localized in both image and camera data. We use the same target here. Figure 4.10 shows the target, which is a truncated pyramid shape with coloured front and back planes to aid segmentation in the image. The target size and position is chosen so that it is easily visible to both sensors when the robot is stationary. The laser scanner is prone to producing so called *mixed measurements* at depth discontinuities (see Figure 4.9), so the target is deliberately constructed to be continuous. Further, all of the edges of the target lie at the intersection of a pair of planes so that the edges may be recovered without explicitly searching for them. Because of this, the procedure can be largely automated, and the user is not expected to perform any tasks requiring high accuracy.



(a) **(b)**

Figure 4.10: The calibration target as seen from the robot's camera (a) and from a more oblique angle (b), showing the truncated pyramid shape.

4.2.2 Our method

The calibration procedure we present now may be divided into two stages. First is to accurately find the 8 corner points of the target in both the image and laser coordinate frames. We do this for data from a number of different robot poses. We attempt to locate the target in the laser data by first classifying the target plane to which each data point belongs. We then iteratively fit a wireframe model of the calibration target to those points. The second stage is to find the transformation $\mathbf{R}, \mathbf{T}$ which minimizes the reprojection error

$$\min_{\mathbf{R},\mathbf{T}} \sum_{i=1}^{n} \|u_i - \hat{u}(\mathbf{R}, \mathbf{T}, \mathbf{X_L})\|^2 \tag{4.11}$$

where u_i are image corner points and $\hat{u}(\mathbf{R}, \mathbf{T}, \mathbf{X_L})$ are the laser points projected into the image, via transformation $\mathbf{R}, \mathbf{T}$ and the intrinsic calibration parameters.

4.2.3 Segmenting the Target from 3D data

RANSAC plane fit

The calibration target is constructed entirely from planes, so after asking the user to roughly crop the point cloud to only include the target, as a first step we iteratively apply a RANSAC [38] plane extraction algorithm. It finds the plane with greatest support from the points, then removes those supporting points from the dataset and finds the next best supported plane, and so on. A point is said to support a plane if its perpendicular distance to that plane is less than some threshold. The threshold must be chosen to be large enough that it accounts for noise in the measurements, but not so large that points from other planes are mistakenly thought to support the current plane. Even with judicious choice of the threshold, there may still be points belonging to another plane which happen to lie within the support threshold of the current plane and will therefore be misclassified.

Figure 4.11 shows the result of applying the iterative RANSAC process to the laser data. The front plane was not successfully extracted by the algorithm because there were not deemed to be enough points to support a plane (many of them having been removed with the larger planes). The RANSAC result *does* however give us a good idea of the dominant plane directions in the dataset. The planes are more robustly classified using a normal-clustering step described next.

Point classification

The final classification of points to specific planes on the target is based on surface normals. The normals of the points are found by fitting a plane through the local neighbourhood of each point, to determine the local orientation of the surface (see Section 2.3). Using a large neighbourhood size ensures that noise is minimized. At this stage, the computed normals may not all be pointing out of the front of the

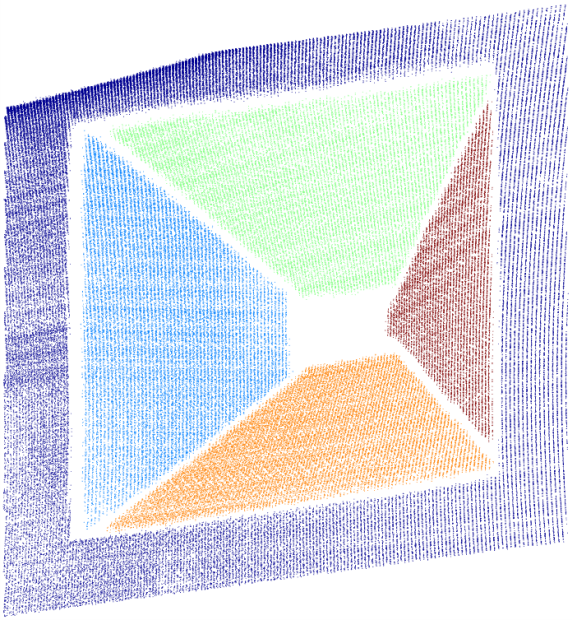


Figure 4.11: Planes found by initial RANSAC extraction. Note how some points that should be on the front plane have been classified as belonging to the inclined plane to the right (red) because they lie within the acceptance threshold of the inclined plane hypothesis. Once points classified as lying on the inclined planes were removed, there were too few points remaining to successfully extract the front plane, which is why it does not appear in the figure. The RANSAC result is however sufficient to give a good initialization to the normals clustering algorithm that follows.

target; some will be pointing backwards. By taking the dot product of the normal with the point to which it belongs, we determine whether the normal is pointing towards or away from the origin. If the latter, then it has its direction reversed.

Figure 4.12 shows a subset of the computed point normals. Because there are 5 dominant plane directions, most normals will be pointing in those directions. The normals of points in noisy areas or close to plane boundaries will be further from the dominant directions.

We project the normals of the points onto the surface of a unit sphere. Figure 4.13 shows that the points are mainly clustered in 5 distinct directions, as expected. The lower-density points forming paths between the clusters are due to laser measurements lying near the boundary of two or more planes. It is these points that we wish to discard. The centres of the 5 clusters are found using the Spherical k-means

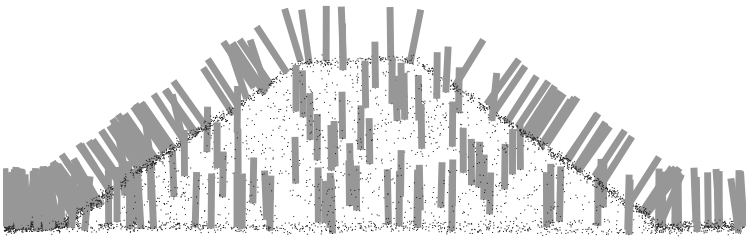


Figure 4.12: A subset of the computed normals of points on the target when viewed from the left hand side. The figure has been rotated 90 degrees to save space.

(`spkmeans`) algorithm of Dhillon and Modha [31], which is detailed in Appendix E. Given an initial guess of the positions of the cluster centres (which we provide from the RANSAC plane fit in Section 4.2.3), the algorithm seeks to iteratively improve the cluster centres until convergence.

The result of the `spkmeans` algorithm is that each point is classified to one of the five cluster centres. We discard all points with normals lying outside of a small threshold distance from their cluster centre – around 5° works well. The remaining points are almost certain to be correctly classified, and will not lie close to plane boundaries on the calibration target. Figure 4.13 shows the cluster centres and final point classification.

Points lying on the four inclined planes of the target are trivially assigned to the correct plane, based on their relative orientation. The front and back planes however have the same orientation, so a further RANSAC plane fitting step is applied to discriminate between the two.

Fitting the Target Model

The result of the previous steps is that for each plane on the target we have a set of relatively noise-free points which belong to the plane. The final step for locating the corner points of the calibration target is an optimization which seeks to best fit a model of the calibration target to the classified point cloud, so that the corner points may be recovered.

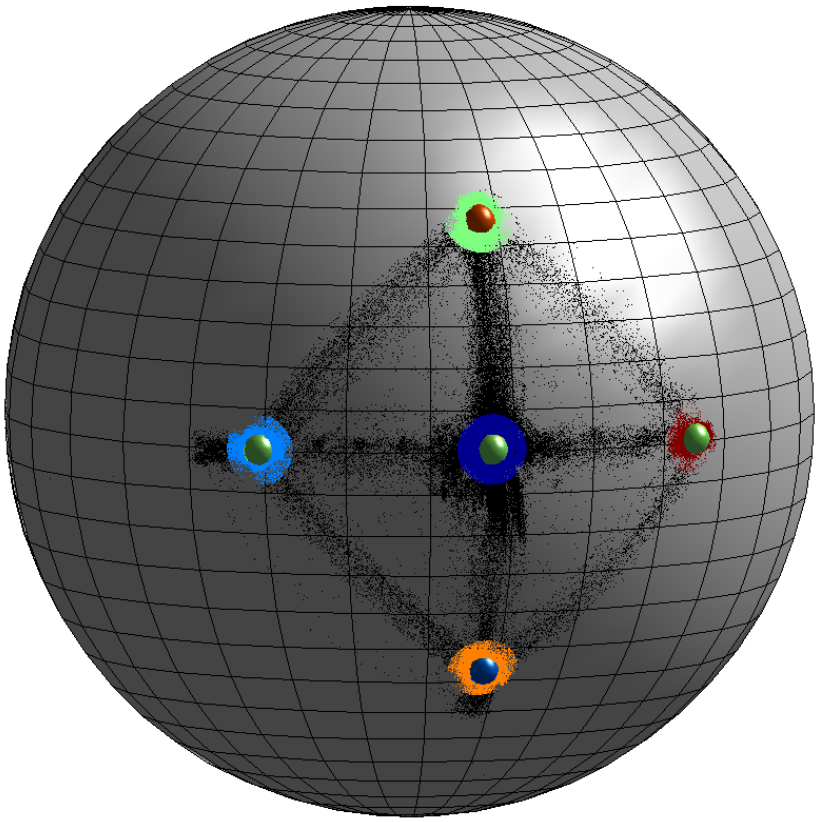


Figure 4.13: Point normals projected on to the surface of a sphere. The 5 dominant plane directions found by RANSAC are represented as small spheres. The normals are each assigned to the closest dominant direction (in a cosine distance sense). Points lying further than a threshold distance from each dominant direction are discarded.

We measured the calibration target very accurately to determine the relative locations of each of the corners. From this, we produced a wireframe model containing the corner locations $\mathbf{P} = \{\mathbf{p_1} \,..\, \mathbf{p_8}\}$ and the set of planes $\mathbf{\Pi} = \{(\mathbf{c}_1, \hat{\mathbf{n}}_1) \,..\, (\mathbf{c}_1, \hat{\mathbf{n}}_1)\}$, where $\mathbf{c}_i$ is some point on the plane, and $\hat{\mathbf{n}}_i$ is the plane's normal. The model is defined relative to an arbitrary local coordinate frame.

From the previous steps, we have for the ith plane the set of laser measurements $\mathbf{\Theta}_i \in X_L$ classified as belonging to that plane. We wish to find the rotation and translation matrices $\mathbf{R_t}, \mathbf{T_t}$ which minimize the expression

$$\underset{\mathbf{R_t}, \mathbf{T_t}}{\arg\min} \sum_{\pi_i \in \mathbf{\Pi}} \sum_{x_j \in \mathbf{\Theta}_i} \left[(\mathbf{R_t} \hat{\mathbf{n}}_i) \cdot (\mathbf{x}_j - \mathbf{R_t} \mathbf{c}_i - \mathbf{T_t}) \right]^2 \tag{4.12}$$

which penalizes the sum of squared orthogonal distances of each measurement from its assigned plane in the model. Because the point-to-plane correspondences are fixed, standard non-linear optimization algorithms such as Levenberg Marquardt converge rapidly to a solution. For the case of our wireframe model, the initial guess need not be particularly close to the true answer because the problem is well conditioned and the basin of convergence is very large.

Once the transformation matrices have been found, they are used to transform the corner points of the wireframe model into the laser frame, giving the locations we need for the extrinsic calibration. Figure 4.14 shows the wireframe model fitted against the classified laser points.

4.2.4 Segmenting the Target in the Image Data

To locate the calibration target's corner points in the camera image we apply the method of Posner et al. [90]. First the image is rectified using the intrinsic calibration parameters of the camera. This ensures that the edges of the calibration target are straight lines in the image. The user is asked to roughly locate the differently

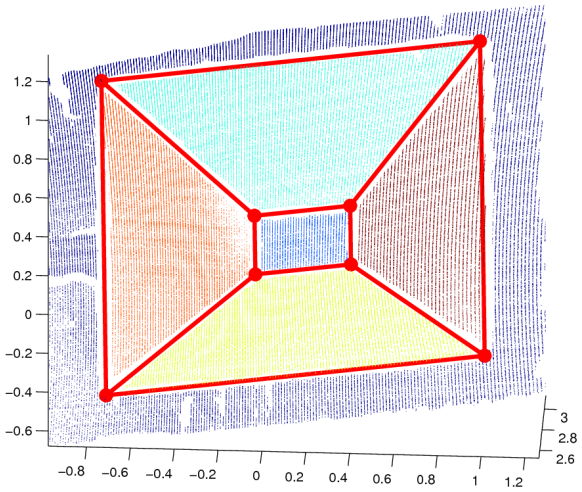


Figure 4.14: The wireframe model of the calibration target is fitted to set of classified laser points. The colour of each point denotes the plane to which it belongs.

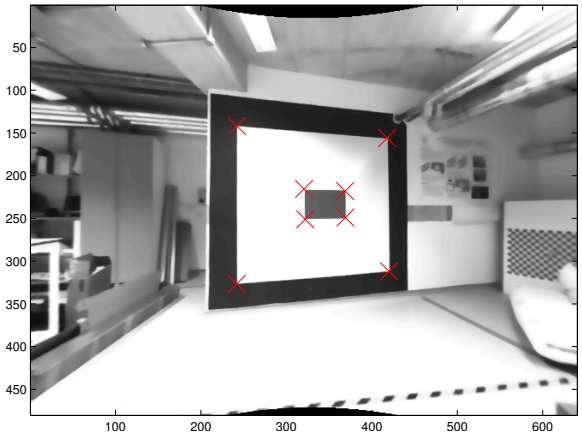


Figure 4.15: Labelled target corner points on the rectified image

shaded parts of the target in the image, and then edge detection is used to locate the rectangles to sub-pixel accuracy. The corners are found at the intersection of the edges, also with sub-pixel accuracy. Figure 4.15 shows a sample image and the detected corners.

4.2.5 Obtaining the Extrinsic Calibration

With the calibration target's corners now located both in the laser data and the camera image, we have eight point correspondences with which to find the extrinsic

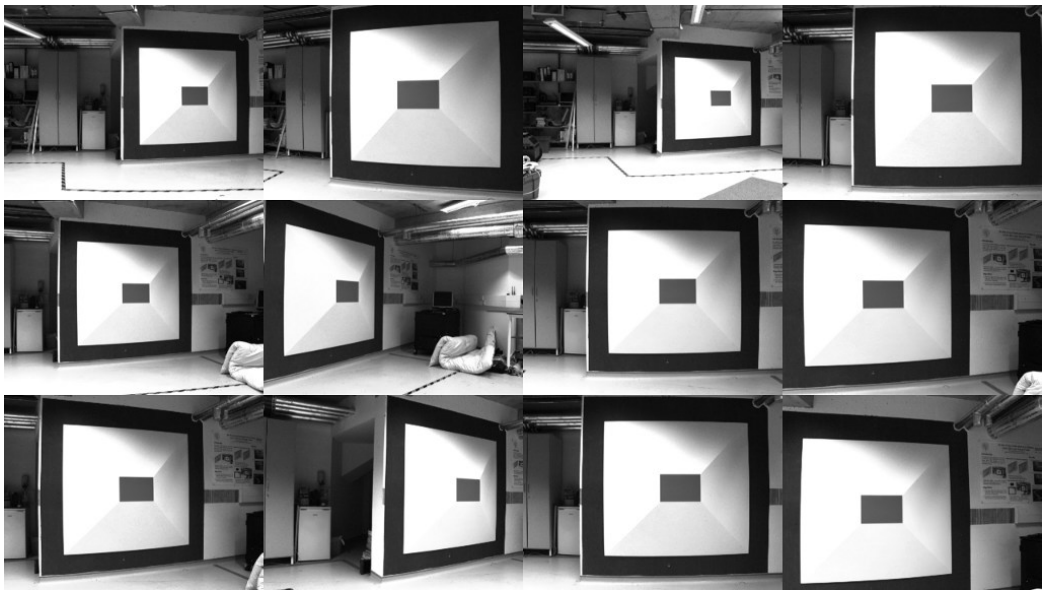


Figure 4.16: We took laser data and camera images of the target from 18 different poses around the target, of which this is a subset. With each pose giving 8 corner correspondences, this resulted in 144 correspondences in total.

transformation matrices, $\mathbf{R}$ and $\mathbf{T}$. Again, we use standard non-linear optimization techniques to minimize the reprojection error in Equation 4.11. The optimization can be seeded from rough measurements of the two sensor's relative locations.

4.2.6 Results

Images and laser data of the target from 18 different poses around the target were taken, to cover as much of the camera's field of view as possible. Figure 4.16 shows the camera images from a subset of the poses. Finding 8 corner correspondences for each using the procedures described resulted in a total of 144 correspondences. We divided the data into a training set of 14 poses (112 correspondences) and a test set of 4 poses (32 correspondences). After solving Equation 4.11 using only the training set, we used the recovered extrinsic parameters to evaluate the errors on the test set. The results are shown in Figure 4.17. The mean error on the test set was $(-0.124, 0.003)$ pixels in x and y. Though individual correspondences found in the data association stages had errors of up to 3 pixels, with enough data the errors cancel each other out

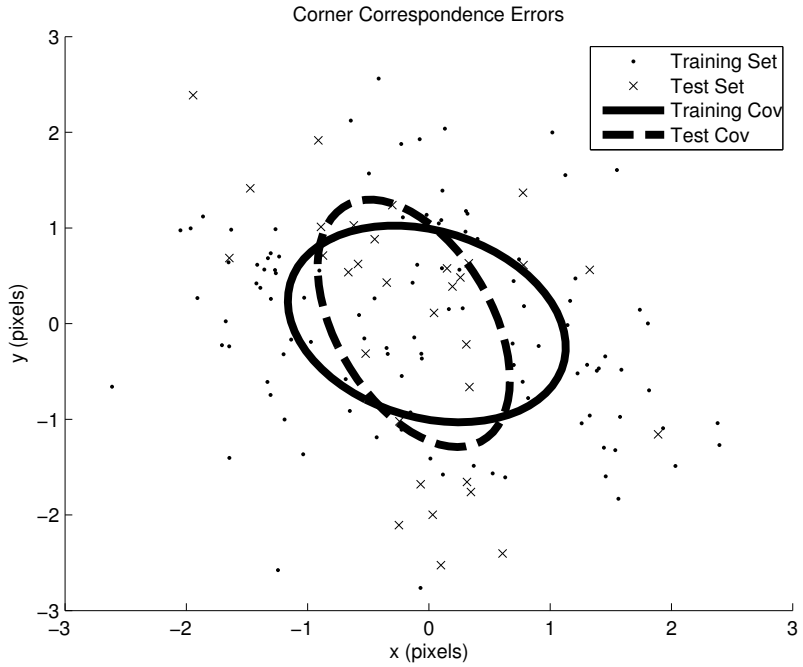


Figure 4.17: Final pixel errors for each of the 144 correspondences. We show both the training set errors and the test set errors here, with their respective 1-sigma covariance bounds. In this case, the mean error of the test set was $(-0.124, 0.003)$ pixels in x and y.

and sub-pixel accuracy is achieved.

With the extrinsic calibration parameters available, it is a simple matter to produce coloured point clouds. The laser points are first transformed by the extrinsic calibration matrices and then projected onto the camera's image plane. The colour of the corresponding pixel is then assigned to each laser measurement. Figure 4.18 shows an example coloured point cloud produced after an extrinsic calibration experiment.

4.3 Chapter Summary

This chapter has described a new method for recovering vehicle trajectories from 3D point cloud data, even when the vehicle's odometry is in error. Basic 2D odometry can be enriched with roll and pitch information which was previously unavailable. The method uses architectural priors in order to overcome the problem of sparse data, which makes standard point cloud registration methods difficult to use effectively.

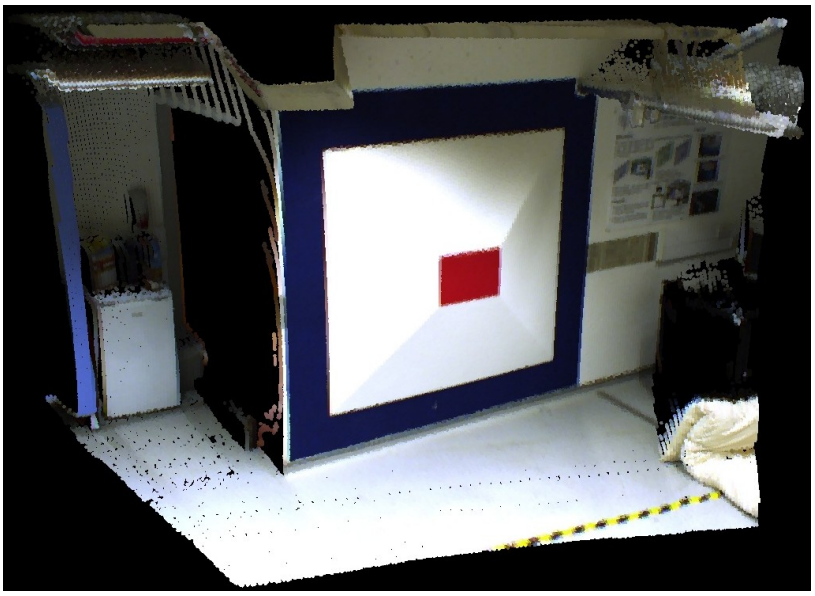


Figure 4.18: Extrinsic Calibration allows us to assign a colour to each point.

The resulting corrected point clouds are suitable for use with scan-matching SLAM algorithms, making rapid map acquisition more feasible.

We have also described the techniques we use to find the extrinsic calibration between a 3D laser scanner and a camera on the same vehicle. Using a specially constructed target, the usually time consuming procedure is largely automated. Robust point classification in laser and image data ensures that the calibration result is of high quality. What has not been addressed in this Chapter and the previous one is an empirical development of a full and realistic sensor model for the laser. There would be a benefit in future work that considers how range measurements are affected not just by the geometry of the scene (angle of incidence), but the material qualities. Indeed, one might consider a sophisticated model that leverages appearance information gleaned from images to enable the construction of a time-varying sensor model. Regretfully this topic is left for future derivative work.

The next chapter continues in a similar vein to this one in that it is about the exploitation of laser range data. In particular it investigates how visual cues can be used in conjunction with laser data to provide dense range images.

Chapter 5

MRF point cloud enhancement

In this chapter we add vision to the mix. We describe a method for fusing low density laser data with high-resolution camera images, to produce dense and accurate point cloud representations of a scene. The motivation for this comes from the fact that a robot can rapidly gather high-resolution images from a low-cost camera, but that accurate range measurements from a laser range sensor tend to be either slow to acquire, or sparse, as well as having no colour information. Note also that we aim to recover the dense geometry of a scene over ranges which prohibit the use of other direct methods such as stereo unless a large baseline is used.

5.1 Existing Work

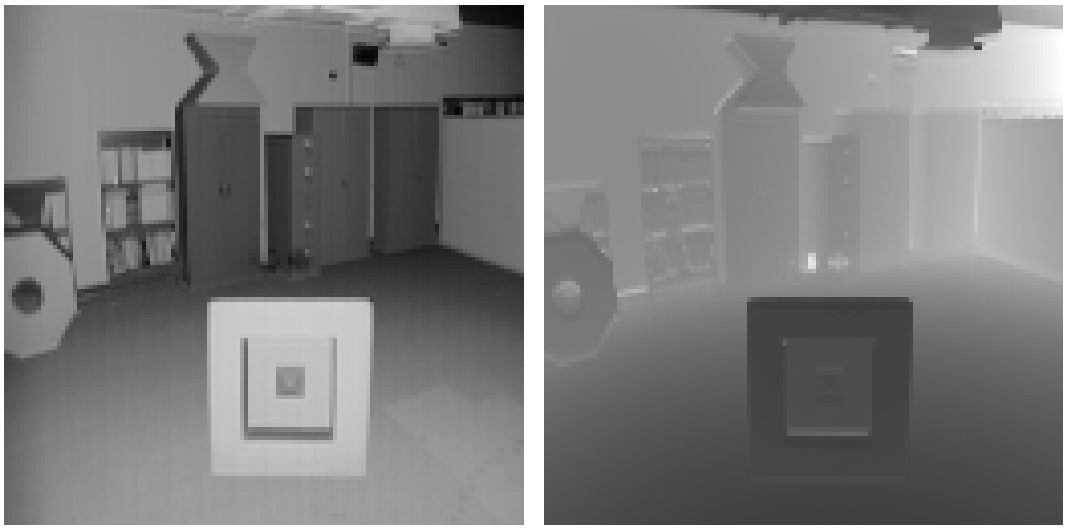
The problem of inferring 3D surface models of a scene using laser or camera sensors has been studied extensively over many years. However, limitations in hardware and a requirement for expeditious data gathering in mobile robotics typically results either in high resolution optical images only allowing inference of very basic 3D geometry, or, alternatively, low resolution range images which often sample the scene too sparsely to allow for faithful reconstruction.

Laser only techniques may be broadly split into three areas. In the first, the 3D

points are used to estimate a signed distance function and then iso-surfacing techniques are employed to extract the zero set of the function to yield the final surface [50, 85]. There are 'crust' methods [2, 58] which compute a Delaunay tetrahedralization of the data points and then attempt to label tetrahedra as lying inside or outside the surface. The interface between the two sets of tetrahedra is the estimated surface. Finally there are volumetric methods which build a surface representation in a voxel grid using techniques such as space carving [29]. All of these methods rely on the surface having been sampled sufficiently densely - usually that neighbouring measurements can be assumed to lie on the same surface. The work described here involves laser measurements an order of magnitude less dense, and somewhat below the spatial Nyquist frequency for the scale of features we wish to reconstruct.

There are methods to estimate geometry or depth from a single image [94, 49, 27], without any metric measurements. They achieve this by relying on strong priors and complex visual cues. The results are compelling, but not yet adequate for accurate dense reconstruction of arbitrary scenes.

Multiple view reconstruction provides an attractive alternative, leading to dense scene reconstructions from image data alone [97, 47]. Unfortunately, stereo reconstruction fidelity is limited in range by the baseline and the image resolution. This impedes accurate reconstruction beyond a few metres from the camera. State-of-the-art graphcut based techniques have until now been limited to using only first order smoothness priors, due to the structure of the stereo reconstruction problem. First order priors favour fronto-parallel surfaces with low curvature and therefore do not accurately model real scenes. Second order priors are recognized to be a more appropriate model, since they allow surfaces of arbitrary orientation and penalize curvature rather than steepness. The main contribution of [120] was the introduction of second order smoothness priors to the stereo problem, but the method is computationally intensive.



(a) Intensity image **(b)** Range image

Figure 5.1: Range and intensity are different modalities, but the images share similar structure. Edges in the intensity image often correspond to discontinuities in range, and areas of similar intensity often correspond to smooth surfaces. (Data supplied by CESAR lab, Oak Ridge National Laboratory, USA)

An alternative can be found in the exploitation of the complementary nature of vision and range sensing. Only a relatively small body of work exists on the inference of surfaces by fusing laser data and camera images. On first inspection, it may not be obvious how colour or intensity information from an image can assist in the estimation of depth[1]. Figure 5.1 shows both a greyscale intensity image and a range image of the same scene. Though they are obtained through different sensor modalities, there are obvious congruities between the two. In particular, edges in the intensity image often correspond to depth discontinuities in the range image. Also, areas of similar intensity tend to be smooth in range. The idea is to exploit these similarities to interpolate intelligently between sparse range measurements.

Torres-Méndez and Dudek [108] employ a Markov Random Field technique to infer range values for pixels in an image. Each pixel is represented as a node with a range value attached. The technique is capable of filling relatively large gaps between

[1]We use 'intensity' and 'colour' interchangeably, depending on whether we are working on a greyscale image or a colour image.

known range data by using the notion that spatially close nodes sharing similar image neighbourhoods are more likely to have similar ranges. The technique requires that the supplied range measurements contain some high density areas from which to seed the solution, and is unable to assign ranges outside of those already in the measurements.

Yang et al. [122] take a filtering approach. First they up-sample a low resolution range map to be the same resolution as the high resolution camera image. Then they generate a 'cost volume' which is a function of pixel location and range. The cost for a given range is a function of its distance from the up-sampled range map. The authors then apply a series of bilateral filters [107] to each range slice of the cost volume. These apply Gaussian smoothing to the cost map in areas which do not correspond to image edges. A final high resolution range map is obtained by choosing the range with lowest cost in the smoothed cost volume, for each pixel. This results in visually pleasing results with smooth depth changes. The technique benefits from regularly sampled range maps which keep the solution from being able to deviate too much from the measurements, as it might in less densely sampled areas.

Andreason et al. [3] use colour information to help cluster 3D points in a scene, in order to improve planar models fitted to the data. More recently, Andreasson et al. [4] have proposed a series of greedy interpolation approaches based on Natural Neighbour Interpolation [101]. We will investigate them shortly. The method of Diebel and Thrun [32] is particularly relevant to our work and formed the motivation for our early adoption of an MRF based approach (see Section 5.4). It employs an MRF formulation with a first-order smoothness prior. The prior favours fronto-parallel surfaces, but does not suffer too greatly from this because the range measurements are sufficiently regular and dense, coming from a special range camera sensor. This 'pins' the estimates to lie near the true surface. We will analyze the technique in more depth shortly. Wong et al. [119] apply 1st-Order MRF based super-resolution to the

problem of mapping mines. Being able to control the lighting and having good priors on the surface reflection properties allows them to reconstruct very high-resolution and accurate surface models.

In our work, we use a similar framework to that of Diebel and Thrun. In contrast to their work, the method presented here is targeted at any combination of commonly available monocular camera and scanning laser. In particular, this requires inference of range measurements based on sparse, inhomogeneous range data, so the smoothness prior must be relied upon over larger areas. This motivates our use of an additional second-order prior, which is novel in the context of image and range data fusion. Unlike multi-view reconstruction approaches, the nature of our problem is such that we are able to make use of high order smoothness terms without difficulty. We further provide a new method of dealing with the non-linearities introduced by the diverging rays of a wide-angle camera, which we are unaware of in other work.

5.2 Background

In this section, we develop the notation that will be used throughout the chapter. The general problem of interpolation consists of estimating the underlying surface that generated a set of sparse point measurements and generating extra samples on that surface. This may be tackled in 3D Cartesian space, but the problem may be more tractably formed for our purposes by posing it in image space.

We represent the problem using an augmented range image representation. Knowing the projection model of the camera and the extrinsic calibration between camera and laser allows us to take a set of laser measurements of a scene and project them into an image of the same scene. Snapping projected points on to the nearest pixel allows those pixels to be augmented with a *known* range measurement.

Figure 5.2 shows how the augmentation occurs and Figure 5.3 shows a camera

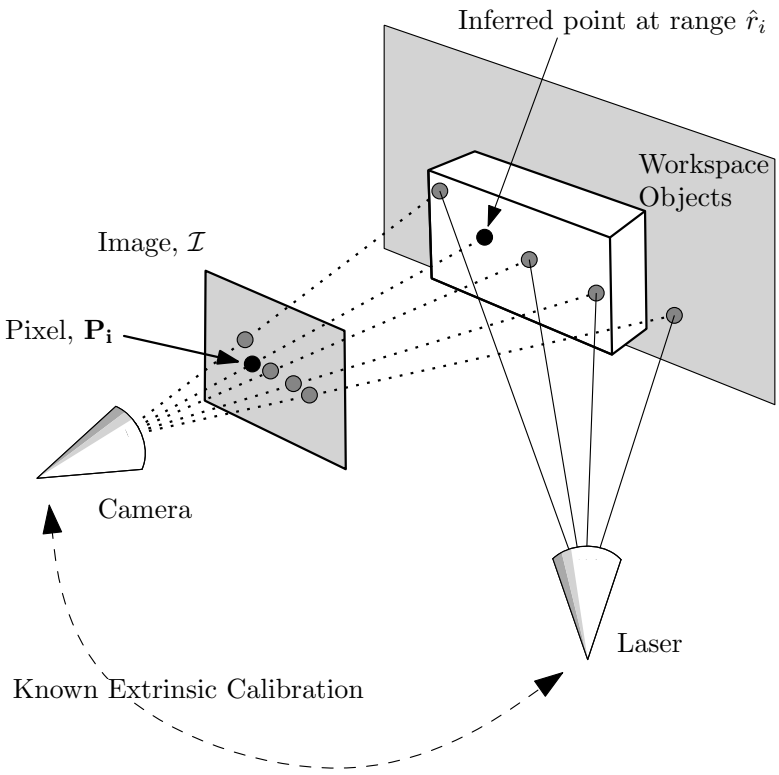


Figure 5.2: The scene reconstruction problem. The camera and laser image the same objects. Some pixels in the image have an associated laser measurement, projected into the image via a known extrinsic calibration. The task is to infer a range for *every* pixel using information in the image and the point cloud.

image with laser points projected in to it. The heart of the problem is how to sensibly infer ranges for pixels which are not near any laser measurements without introducing intolerable distortions. Our method is general in that it is not tied to any particular 3D laser scanner mechanism or geometry.

We use similar notation to Andreasson et al. [4]. If $\mathcal{I}$ is a colour image of dimension $h \times w$, then each pixel may be represented as $\mathbf{P_j} = (X_j, Y_j, C_j)$ for $j = 1..N$, where $C_j = (C_j^R, C_j^G, C_j^B)$ is an RGB colour vector and $N = h \times w$. Suppose we have a set of laser range measurements, $\mathbf{r}$, with a laser range measurement $r_i \in \mathbf{r}$ which projects onto pixel $\mathbf{P_i}$ then we can augment the pixel with the range to give $\mathbf{R_i} = (X_i, Y_i, r_i, C_i)$. These augmented pixels are known as *measured nodes*. For a pixel $\mathbf{P_j}$ which has no associated laser range, we must interpolate from measured nodes. A node with an *estimated* range is written as $\hat{\mathbf{R}}_\mathbf{j} = (X_j, Y_j, \hat{r}_j, C_j)$ and called

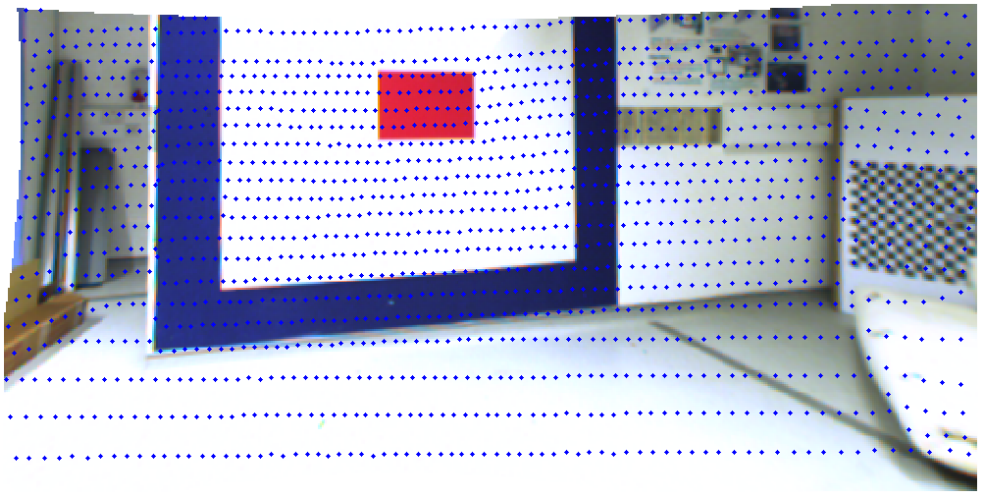


Figure 5.3: A rectified image with laser measurements superimposed. Note the sparseness of the laser data, particularly on the floor, where the stripes are ~ 0.5m apart.

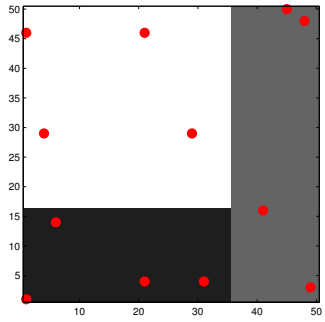
an *estimate node*.

5.3 A closer look at some existing methods

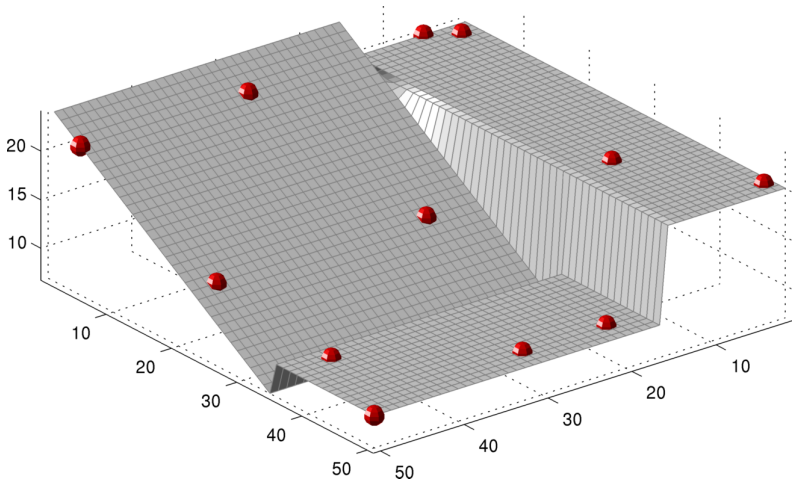
In this section, we describe and analyze the approaches of Andreasson et al [4], and of Diebel and Thrun [32]. We will use two synthetic problems to demonstrate the general behaviour of each.

5.3.1 Greedy Linear Interpolation

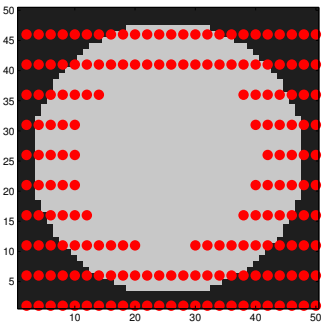
Andreasson et al. [5] present five non-iterative linear interpolation algorithms to solve the range data and image fusion problem. We have implemented all five, to use as benchmarks for our own algorithm, so they are described here. All five operate by looping through all estimate nodes and assigning them a range which is a function of only the measured nodes. The algorithms are greedy because they never revisit a range estimate once it has been assigned.



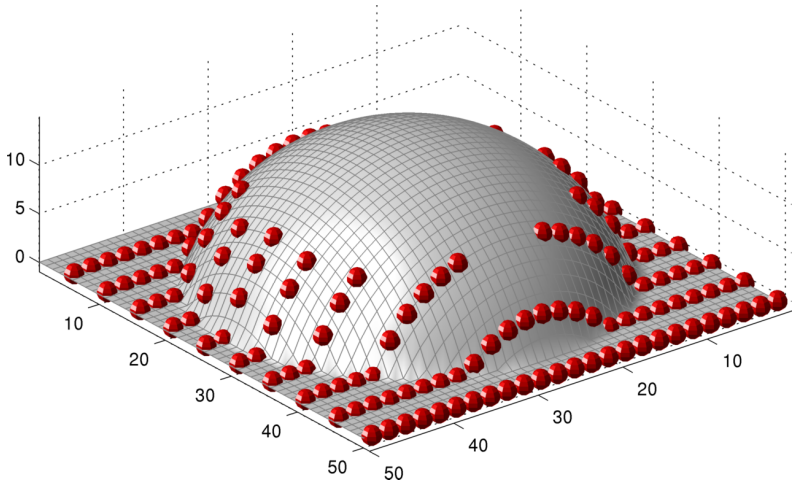
(a) Three Planes Image



(b) Three Planes Ground Truth



(c) Dome Image



(d) Dome Ground Truth

Figure 5.4: Synthetic data examples which highlight important aspects of each approach, with ground truth surfaces. Each intersection in the mesh represents a single range node projected out from the corresponding image pixel. The images for each of the two cases are shown on the left. Sparse laser measurements are shown in red.

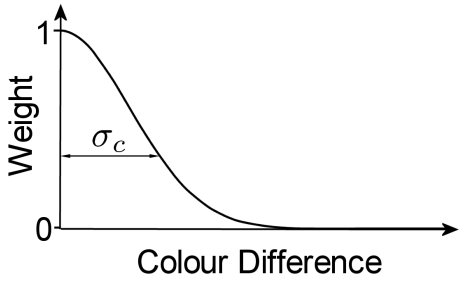


Figure 5.5: Variation of the colour weighting term with pixel colour difference.

Nearest Range Reading (NR)

The first, known as *Nearest Range Reading* simply assigns to each estimate node the range of the closest (in pixel distance) measured node. The effect of this is that the Voronoi region around each measured node becomes a fronto-parallel planar patch. The probabilistic interpretation is that range $\hat{r}_j$ is chosen to maximize the likelihood

$$p(\mathbf{P_j}, \mathbf{R_i}) \propto e^{-\frac{(X_j - X_i)^2 + (Y_j - Y_i)^2}{\sigma_d^2}} \tag{5.1}$$

where σ_d is a variance for pixel distance and $\mathbf{R_i}$ is a measured range node. In this algorithm it has no effect on the result.

Nearest Range Reading Considering Colour (NRC)

More interesting is a method which uses colour information in addition to distance. Range $\hat{r}_j$ is chosen to maximize the likelihood

$$p(\mathbf{P_j}, \mathbf{R_i}) \propto e^{-\frac{(X_j - X_i)^2 + (Y_j - Y_i)^2}{\sigma_d^2} - \frac{\|C_j - C_i\|^2}{\sigma_c^2}} \tag{5.2}$$

where σ_d is a variance for pixel distance and σ_c is a variance for colour difference. If the closest measured node has a different colour than the estimate node being considered, then the algorithm may assign the range of a different measured node a little further away. The two variances are chosen to tune the behaviour of the algorithm. Figure

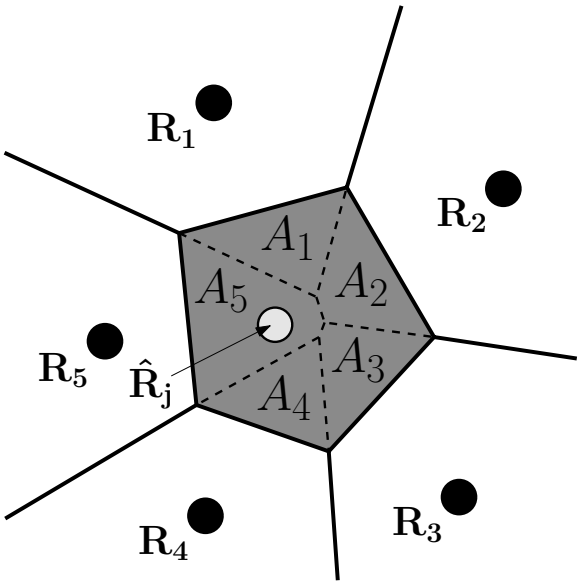


Figure 5.6: The dashed lines show the original Voronoi diagram of nodes $\mathbf{R_1}..\mathbf{R_5}$. The shaded area is the Voronoi region of estimate node $\hat{\mathbf{R}}_\mathbf{j}$ after it is added. $\mathbf{R_1}..\mathbf{R_5}$ are said to be Natural Neighbours of $\hat{\mathbf{R}}_\mathbf{j}$, because it 'steals' area from their Voronoi regions. In Natural Neighbour Interpolation, the summation weight of each measured node $\mathbf{R_i}$ is proportional to the area A_i stolen from it by $\hat{\mathbf{R}}_\mathbf{j}$

.

5.5 shows how the colour weight varies with pixel difference.

Multi-Linear Interpolation (MLI)

This uses the *Natural Neighbours* interpolation method of Sibson [101]. Consider Figure 5.6, which shows measured nodes $\mathbf{R_1}..\mathbf{R_5}$, and their Voronoi diagram. When the estimate node $\hat{\mathbf{R}}_\mathbf{j}$ is added and the Voronoi diagram updated, the Voronoi regions of $\mathbf{R_1}..\mathbf{R_5}$ all have to be truncated to make way for the new Voronoi region. This makes the nodes $\mathbf{R_1}..\mathbf{R_5}$ the *Natural Neighbours* of $\hat{\mathbf{R}}_\mathbf{j}$.

Natural Neighbour Interpolation can be used to provide a range estimate for $\hat{\mathbf{R}}_\mathbf{j}$ by a weighted sum of measured ranges,

$$\hat{r}_j = \sum_{i \in \mathrm{NN}(\hat{\mathbf{R}}_\mathbf{j})} w_i(\hat{\mathbf{R}}_\mathbf{j}) r_i \tag{5.3}$$

where $\mathrm{NN}(\hat{r}_j)$ is an operator selecting all the Natural Neighbours of $\hat{\mathbf{R}}_\mathbf{j}$ and w_i is the

weight of each measured range:

$$w_i(\hat{\mathbf{R}}_\mathbf{j}) = \frac{A_i}{A} \tag{5.4}$$

A_i is the area 'stolen' from the Voronoi region of $\mathbf{R_i}$ by $\hat{\mathbf{R}}_\mathbf{j}$, and A is the total area stolen, so that $A = \sum_i A_i$. Natural Neighbour Interpolation leads to smooth and visually pleasing surfaces.

Multi-Linear Interpolation Considering Colour (LIC)

This is a method of fusing colour information into Natural Neighbour Interpolation. Another weighting term is introduced, to represent the colour difference between an estimate node and its Natural Neighbour measured nodes:

$$w_i^c(\hat{\mathbf{R}}_\mathbf{j}) = e^{-\frac{\|C_i - C_j\|^2}{\sigma_c^2}} \tag{5.5}$$

The interpolated range becomes,

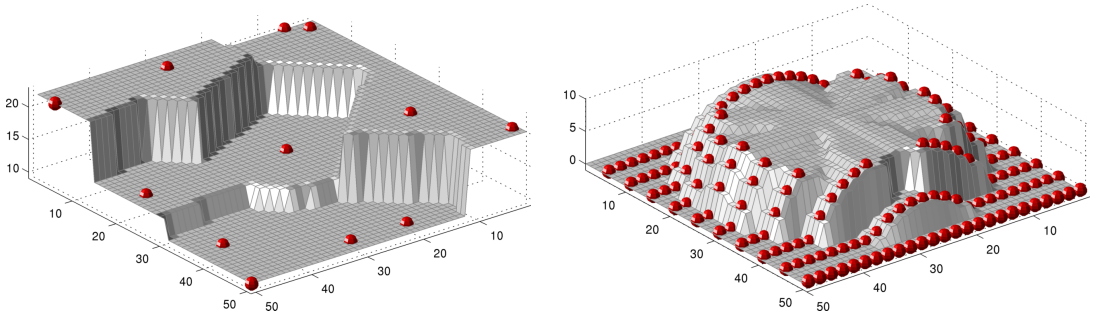
$$\hat{r}_j = \sum_{i \in \mathrm{NN}(\hat{\mathbf{R}}_\mathbf{j})} \frac{w_i w_i^c}{W} r_i \tag{5.6}$$

with normalization factor $W = \sum_i w_i w_i^c$.

Results on the synthetic datasets for the four methods described are shown in Figure 5.7.

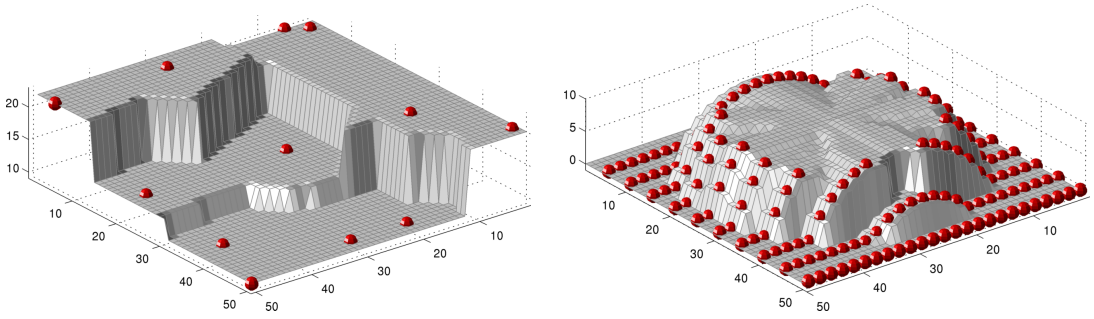
Parameter-Free Multi-Linear Interpolation Considering Colour (PLIC)

The final algorithm is an attempt to do away with the tuning parameter, σ_c in the LIC algorithm. The range interpolation function is identical to that in LIC, except



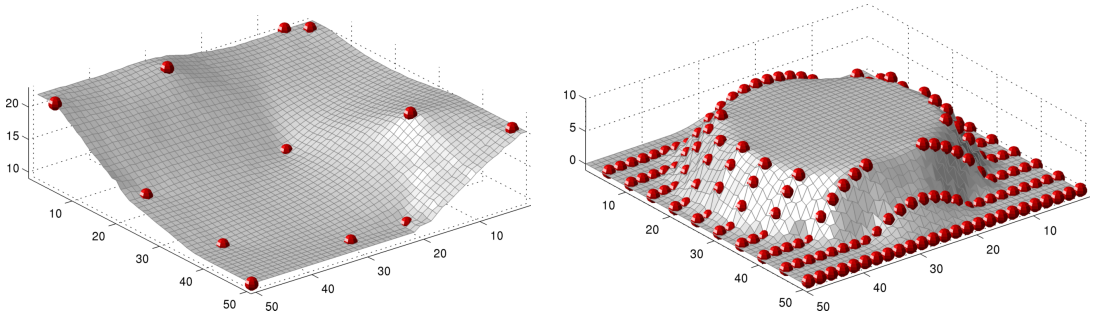
(a) Three Planes, NR

(b) Dome, NR



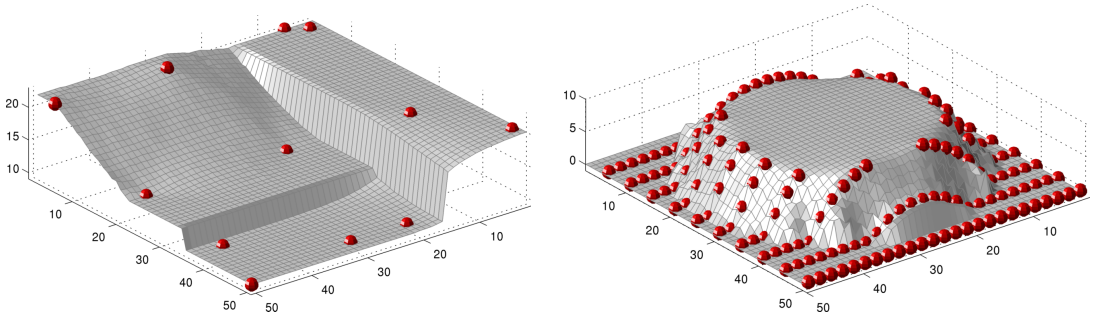
(c) Three Planes, NRC ($\sigma_c = 2.7$)

(d) Dome, NRC ($\sigma_c = 2.7$)



(e) Three Planes, MLI

(f) Dome, MLI



(g) Three Planes, LIC ($\sigma_c = 0.3$)

(h) Dome, LIC ($\sigma_c = 0.3$)

Figure 5.7: Results for the first four methods of Andreasson et al. when they are applied to the synthetic cases of Figure 5.4. We have not shown the results for PLIC, because on these examples they closely match LIC. LIC clearly out-performs the others, but it still favours flat surfaces, as evidenced by the flat area at the bottom of the inclined plane. None of the methods is able to correctly reconstruct the Dome.

that σ_c is adaptively computed for each natural neighbour by the equation,

$$\sigma_c^2 = \frac{1}{n_i - 1} \sum_{j \in A_i} \|\mu_i - C_j\|^2, \tag{5.7}$$

which is the colour variance of the pixels inside the stolen area for $\mathbf{R_i}$. n_i is the number of pixels inside A_i and $\mu_i = (1/n_i) \sum_{j \in A_i} C_j$ is the mean colour in A_i. The intuition is that natural neighbours having stolen regions with large colour variance will cause the colour similarity term to have less effect on the weight in Equation 5.6.

In practice we have found that the PLIC method often does not choose a good value for σ_c. Choosing the value well depends on the colour variance of a region in the image. When measured nodes are close together, the number of pixels falling within the stolen area may be very small - sometimes fewer than five. Calculating a variance value for so few samples gives a very weak estimate, so the algorithm suffers. The suprising side-effect of this is that to a certain extent the algorithm actually performs better when the Measured Nodes are less dense in the image.

5.3.2 First-Order MRF Approach

We now turn our attention to the Markov Random Field (MRF) based method of Diebel and Thrun [32]. Unlike the schemes of Andreasson et al., the output surface does not pass through the known measurements. Instead the method seeks to infer a range for *every* node such that a cost function is minimized. They define two energy terms of the form:

$$\textbf{Data Cost:} \qquad \Psi = \lambda^{dat} \sum_{i \in \mathbf{r}} (\hat{r}_i - r_i)^2 \tag{5.8}$$

$$\textbf{1st-Order Smoothness Cost:} \qquad \Phi^{1st} = \lambda^{1st} \sum_{j=1}^{N} \sum_{i \in \mathrm{Nb}(j)} w_{i,j} (\hat{r}_i - \hat{r}_j)^2, \tag{5.9}$$

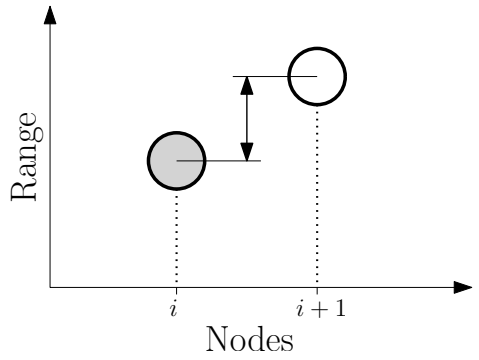


Figure 5.8: The 1st-Order Smoothness cost penalizes neighbouring nodes having different ranges. It tries to make all nodes have the same range.

where Nb(j) is a function selecting the immediate horizontal and vertical neighbours of node $\hat{\mathbf{R}}_\mathbf{j}$ and

$$w_{i,j} = \exp\left(-\frac{\|C_i - C_j\|^2}{\sigma_c^2}\right) \tag{5.10}$$

is a weighting term based on the colour difference between pixels $\mathbf{P_i}$ and $\mathbf{P_j}$, much like Equation 5.5 for Andreasson's LIC method. λ^{dat} and λ^{1st} are scalar parameters representing the relative importance of the data cost and the smoothness cost[2].

The Data Cost acts only on nodes that have a measured range attached to them. It penalizes the distance between the measured range and the inferred range. The Smoothness Cost tries to ensure that adjacent nodes have similar ranges by penalizing differences between them (Figure 5.8). But crucially the weighting term $w_{i,j}$ is chosen such that the contribution of a pair of nodes is reduced when they have dissimilar pixel colours. This has the effect of explicitly allowing range discontinuities over edges in the image. This is what gives the technique its power.

The two energy terms are combined into a conditional distribution over range

[2]Note we have changed Diebel and Thrun's notation to bring it in line with our own notation scheme.

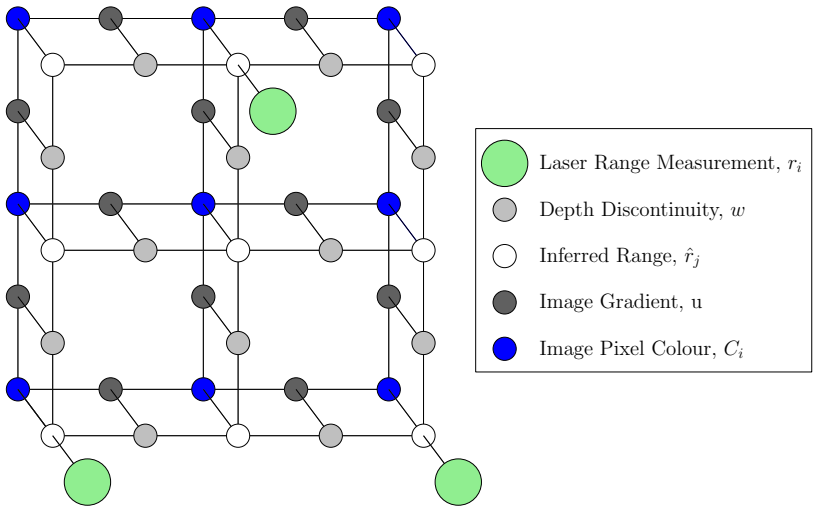


Figure 5.9: The MRF structure proposed by Diebel and Thrun. The Inferred Range nodes are those which the MRF is attempting to infer from all of the other data. Laser range measurements are sparse compared to pixels, so only some pixel nodes have an attached range measurement node.

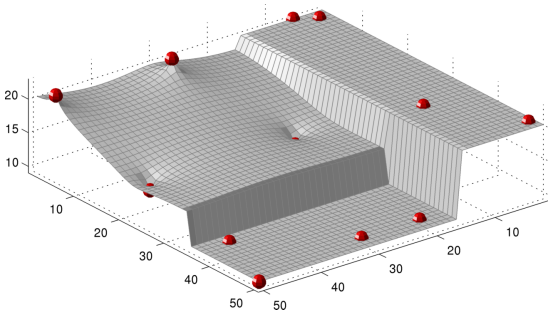
values, given the image $\mathcal{I}$ and the known range measurements, $\mathbf{r}$:

$$p(\hat{\mathbf{r}} \mid \mathcal{I}, \mathbf{r}) \propto \exp\left[-\frac{1}{2}(\Psi + \Phi^{1st})\right] \tag{5.11}$$

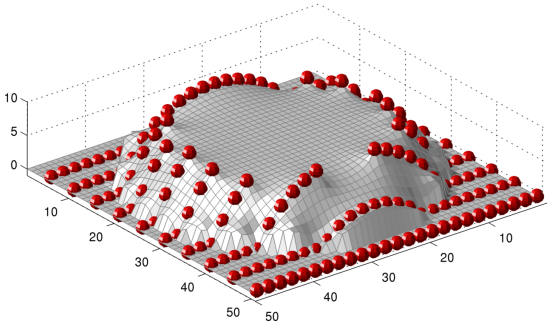
The goal is to find the set of range values, $r_1 \ldots r_N$ (one for each pixel node) which maximize the posterior. This is equivalent to minimizing the expression $\Psi + \Phi^{1st}$, which is a least-squares optimization problem that Diebel and Thrun solve using the Conjugate Gradient method [91]. As an initial guess they simply use a linear interpolation through the known range measurements, ignoring colour information. The graphical representation of the problem is shown in Figure 5.9.

Applying the 1st-Order MRF to the Synthetic Cases

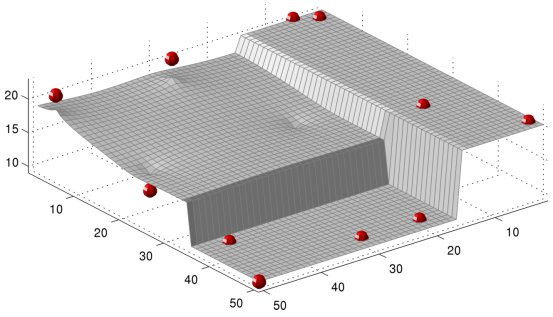
Figure 5.10 shows the results of applying the 1st-Order MRF method to the synthetic cases, and the tradeoff between the data and smoothness terms. The Smoothness term has a strong affinity for flat planes, as it strongly penalizes adjacent nodes with different ranges. Note the 'puckering' effect near to the data points, where the Data



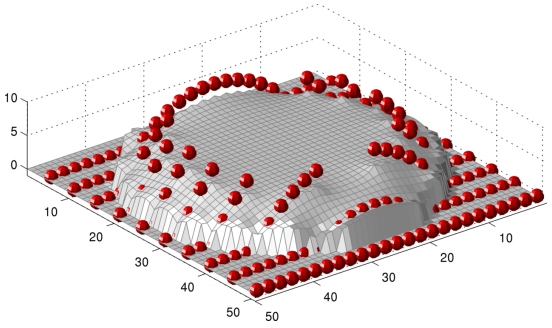
(a) Three Planes, Mostly Data
$\lambda^{dat} = 0.99, \lambda^{1st} = 0.01$



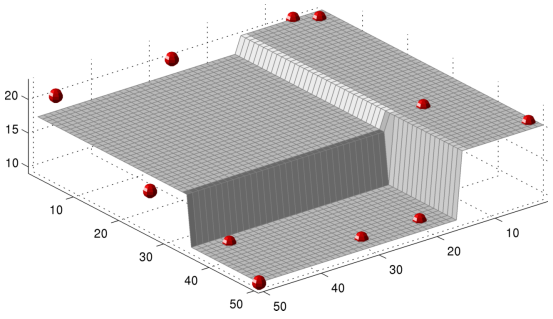
(b) Dome, Mostly Data
$\lambda^{dat} = 0.99, \lambda^{1st} = 0.01$



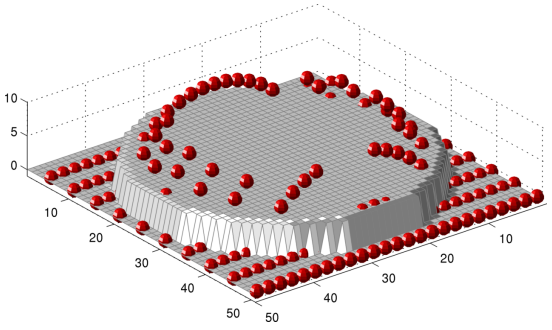
(c) Three Planes, Equal Data and Smoothness
$\lambda^{dat} = 0.42, \lambda^{1st} = 0.58$



(d) Dome, Equal Data and Smoothness
$\lambda^{dat} = 0.57, \lambda^{1st} = 0.43$



(e) Three Planes, Mostly Smoothness
$\lambda^{dat} = 0.05, \lambda^{1st} = 0.95$



(f) Dome, Mostly Smoothness
$\lambda^{dat} = 0.05, \lambda^{1st} = 0.95$

Figure 5.10: Results on the synthetic datasets for the 1st-Order MRF technique of Diebel and Thrun. The relative weighting between Data and Smoothness has been varied to show their different effects. In all cases, we set the colour variance term $\sigma_c = 1.0$. When the smoothness term greatly outweighs the data term, the surfaces become flat and pass through the mean of the data points within regions of similar colour. The method still allows large discontinuities across image edges.

term is able to overcome the Smoothness term.

5.4 Adding a 2nd-Order Smoothness Term

We shall now describe our main contribution, which is the addition of a 2nd-Order smoothness term to the MRF framework. In contrast to Diebel and Thrun [32] we make the further assumption that in the absence of cues to the contrary, such as discontinuities in appearance, the gradient of surfaces varies smoothly. This leads to a method which does not favour fronto-parallel planes, because it no longer penalizes adjacent nodes having different range values. It allows arbitrarily inclined planes without extra penalty. Instead, this *2nd-Order* term penalizes curvature. Data errors are distributed across the surface, resulting in a minimum-energy curved surface, so that cases such as our 'Dome' example are more faithfully reconstructed. We use the same formulation as Diebel and Thrun, except that we add an extra energy term:

$$p(r \mid \mathcal{I}, \mathbf{r}) \propto \exp\left[-\frac{1}{2}(\Psi + \Phi^{1st} + \Phi^{2nd})\right] \tag{5.12}$$

The rest of this section will be devoted to determining a suitable form for the 2nd-Order smoothness term which penalizes curvature rather than slope. For a given node $\hat{\mathbf{R}}_\mathbf{i}$ we require an expression which results in a scalar value quantifying the curvature of the close neighbourhood of $\hat{\mathbf{R}}_\mathbf{i}$. In order to maintain the Linear Least-Squares form of the cost function, the expression must be a linear combination of neighbouring node values. For simplicity we consider a 1D case here, but the method is perfectly applicable to both horizontal and vertical neighbourhoods.

5.4.1 Curvature measurement

Consider Figure 5.11a, which shows $\hat{\mathbf{R}}_\mathbf{i}$ and its horizontal 5-neighbourhood. The general form of the curvature cost will be

$$\Phi^{2nd} = \lambda^{2nd} \sum_{i=1}^{N} (\hat{r}_i^* - \hat{r}_i)^2 \tag{5.13}$$

$$\text{with} \qquad \hat{r}_i^* = \sum_{j \in \text{Nb}(i)} w_{i,j}^n \hat{r}_j \tag{5.14}$$

where this time the Nb(i) function considers a slightly larger neighbourhood of node $\hat{\mathbf{R}}_\mathbf{i}$. λ^{2nd} is a scalar term which sets the relative weight of the 2nd-Order smoothness and $w_{i,j}^n$ are the relative contributions of each neighbouring node to the curvature measure. We will discuss how these are chosen next. Think of $\hat{r}_i^*$ as being a prediction of the minimum curvature location for the ith node, given only the location of its neighbours.

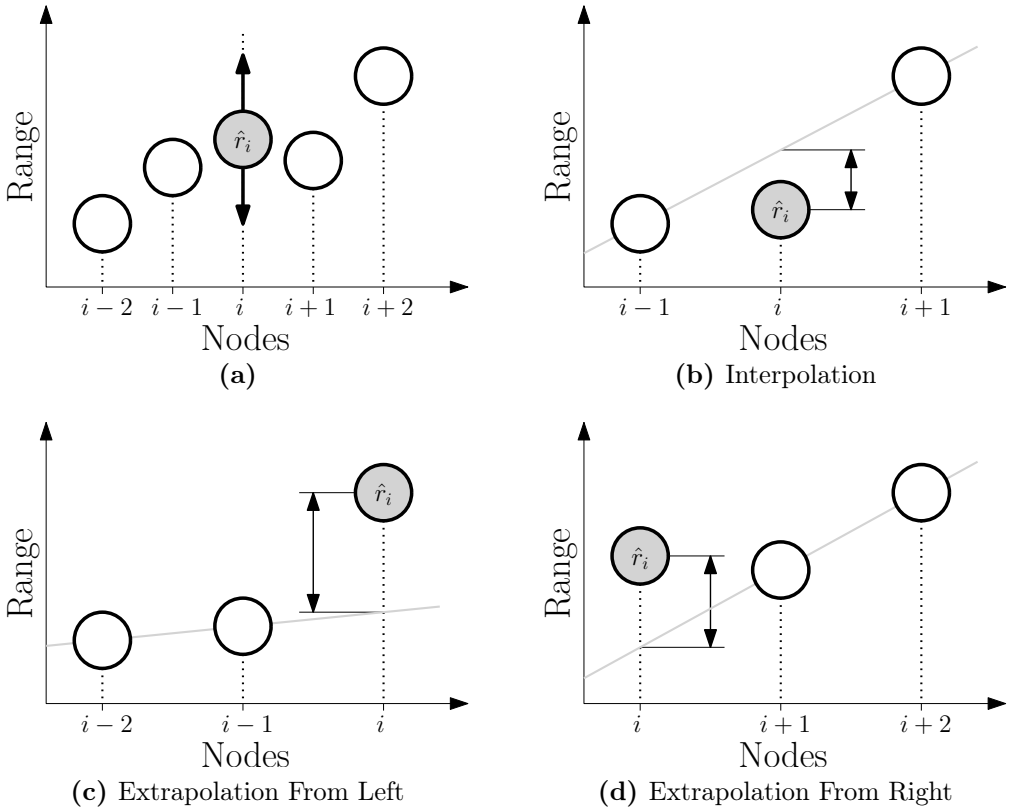
There are three ways in which we choose to derive a curvature measure for the neighbourhood; shown in Figure 5.11, they are:

Interpolation A line is fitted between the two closest neighbours of $\mathbf{R}_\mathbf{i}$ and the distance between the line and $\hat{r}_i$ is penalized. In the 1D case,

$$\hat{r}_i^{*(Interp)} = 0.5\hat{r}_{i-1} + 0.5\hat{r}_{i+1} \tag{5.15}$$

Left Extrapolation As before, but a line is fitted between the two left-hand neighbours of $\mathbf{R}_\mathbf{i}$ so that

$$\hat{r}_i^{*(LExtrap)} = -1.0\hat{r}_{i-2} + 2.0\hat{r}_{i-1} \tag{5.16}$$



(a) **(b)** Interpolation

(c) Extrapolation From Left **(d)** Extrapolation From Right

Figure 5.11: The top-left figure shows an example 1D neighbourhood of a node $\mathbf{R_i}$, posing the question: "What is the range of the middle node that minimizes the curvature of the neighbourhood?" The remaining figures show three possible ways of quantifying curvature, by measuring the distance of the central node $\mathbf{R_i}$ to a line fitted between two neighbouring nodes. Note that the three cases do not represent the same node configuration as the first figure.

Right Extrapolation Mirror of Left Extrapolation, so that

$$\hat{r}_i^{*(RExtrap)} = 2.0\hat{r}_{i+1} - 1.0\hat{r}_{i+2} \tag{5.17}$$

5.4.2 Choosing Which Curvature Measure to Use

The image $\mathcal{I}$ can be used to provide cues about the behaviour of the surface we hope to reconstruct. Our basic assumption is as before — sharp changes in range tend to appear as changes in appearance (edges) in an image. Figure 5.12 demonstrates the idea, which uses the notion that information should not flow between nodes with

dissimilar pixel colour. Broadly speaking, if a pixel is identical to its left and right neighbours then pure interpolation will occur. If however there is a discontinuity in pixel appearance then interpolation will be down weighted and either left or right extrapolation emphasized. In the first case of Figure 5.12, only the Interpolation Measure is valid, because no information can flow from the far left or right across the sharp pixel edges. In the second case, softening the left hand edge allows some information to flow, so that the Left Extrapolation Measure is able to contribute to the prediction of where the ith node should be to minimize the local curvature.

The compound expression for the ith node's prediction is therefore modified to become

$$\begin{aligned}\hat{r}_i^* = \quad & w_i^{Interp}(0.5\hat{r}_{i-1} + 0.5\hat{r}_{i+1}) \\ & + w_i^{LExtrap}(-1.0\hat{r}_{i-2} + 2.0\hat{r}_{i-1}) \\ & + w_i^{RExtrap}(2.0\hat{r}_{i+1} - 1.0\hat{r}_{i+2}),\end{aligned} \tag{5.18}$$

with each Curvature Measure weight based on the pixel colours of the neighbourhood. Interpolation is preferable to extrapolation, so with this preference in mind and considering node $\mathbf{R_i}$, we choose the importance weights of left / right extrapolation and interpolation as

$$\begin{aligned}w_i^{Interp} &= w_{i,i-1}w_{i,i+1} \\ w_i^{LExtrap} &= w_{i+2,i+1}w_{i+1,i}(1 - w_i^{Interp}) \\ w_i^{RExtrap} &= w_{i-2,i-1}w_{i-1,i}(1 - w_i^{Interp}).\end{aligned} \tag{5.19}$$

with $w_{i,j}$ as defined as in Equation 5.10, and shown in Figure 5.13.

The above relationships can be understood by noting that if the pixel attached to range node $\hat{\mathbf{R}}_i$ is identical to its neighbours ($w_{(i-1,i)}$ and $w_{(i,i+1)}$ are unity) then

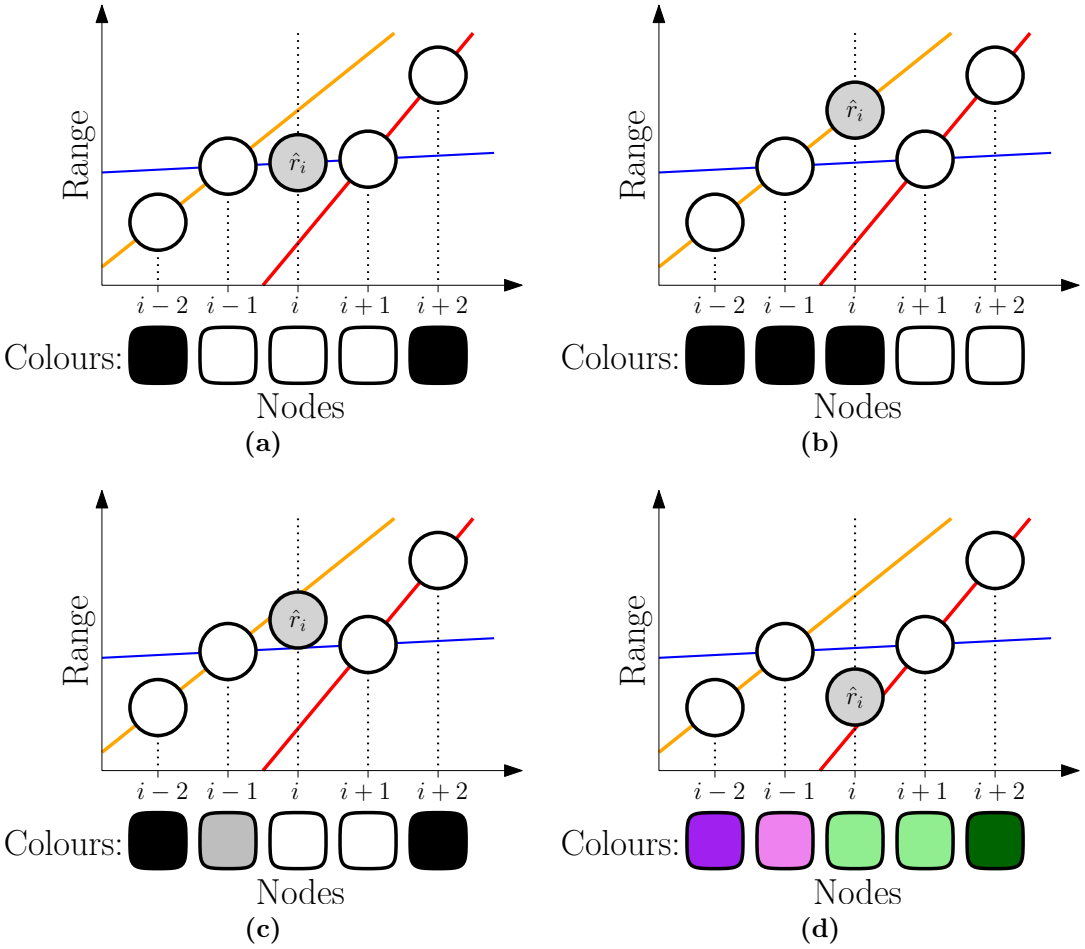


Figure 5.12: The colour of the underlying pixels influences how much weight each curvature measure is given. Pixels with dissimilar colours do not allow information to flow. In the first example no information flows from the two outer nodes, so only the Interpolation Measure is used. In the second figure, information can only flow from the two left-hand nodes, so the Left Extrapolation measure is used. In the third example, some information is able to flow from the far left, so both the Interpolation Measure and the Left Extrapolation Measure are used in equal quantities, but the Right Extrapolation still plays no part. The last example shows a 'softer' case, where all three measures are combined, though there is bias towards Right Extrapolation.

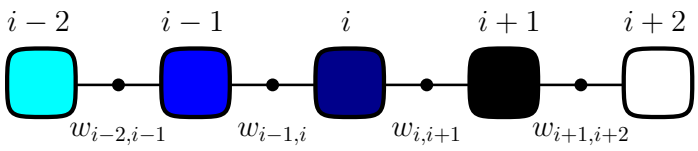


Figure 5.13: A 1D chain of range nodes and the edges between neighbours. Considering the ith node, right extrapolation uses only nodes to the right and left extrapolation uses the two left hand nodes. Interpolation uses the two immediate neighbours. The edges between nodes are a function of the difference in pixel appearance between adjacent range nodes (each range node is associated with a single pixel in the Image).

$w_i^{Interp} = 1$ and $w_i^{RExtrap} = w_i^{LExtrap} = 0$ - interpolation has 100% of the weighting. As the pixels at $\hat{\mathbf{R}}_{i-1}$ and $\hat{\mathbf{R}}_{i+1}$ become increasingly different, the left and right extrapolations receive more weight. In the limit, if two pixels are entirely different, the edge between them tends to zero and the attached range nodes will have no direct link between them. From the perspective of the graphical model this is akin to removing an edge between nodes. It does not make the two nodes independent - there may be other dependencies via long circuitous routes through other nodes. It does however mean that range discontinuities across this boundary are not penalized because the range prediction is based on an extrapolation from one side and not an interpolation across the discontinuity. This is a key point in this work.

5.4.3 A Matrix Representation Including 2nd-Order Terms

We now show how to write the cost function 5.12 in matrix form, so that the optimization problem may be solved using modern sparse linear solve techniques. Recall that the data cost is given by

$$\Psi = \lambda^{dat} \sum_{i \in \mathbf{r}} (\hat{r}_i - r_i)^2 \tag{5.20}$$

where $\hat{r}_i$ is an inferred range, and r_i is a measured range. This may be written as

$$\Psi = \lambda^{dat} \|\mathbf{D}(\hat{\mathbf{r}} - \mathbf{z})\|^2 \tag{5.21}$$

where z is a vector of length N, which contains non-zero range values, r_i only for those nodes that have a known measurement. The remaining elements are set to 0. $\mathbf{D}$ is a Diagonal matrix, with element $\mathbf{D}_{i,i} = 1$ if r_i has a valid range measurement or 0 otherwise. Optionally, we may incorporate measurement uncertainty by setting $\mathbf{D}_{i,i} = \sigma_i$, where σ_i is our confidence in measurement r_i.

The 1st-Order smoothness term can equally be written as

$$\Phi^{1st} = \lambda^{1st} \sum_{j=1}^{N} \sum_{i \in \mathrm{Nb}(j)} w_{i,j} (\hat{r}_i - \hat{r}_j)^2 \tag{5.22}$$

$$= \lambda^{1st} \|\mathbf{F}\hat{\mathbf{r}}\|^2, \tag{5.23}$$

with each row of $\mathbf{F}$ representing a weighted average of a pair of adjacent range nodes. In the 1D case it has values on the diagonal and one off-diagonal. For 2D problems the structure is more complex.

Finally, the 2nd-Order smoothness term is written as

$$\Phi^{2nd} = \lambda^{2nd} \sum_{i=1}^{N} (\hat{r}_i^* - \hat{r}_i)^2 \tag{5.24}$$

$$= \lambda^{2nd} \|\hat{\mathbf{r}}^* - \mathbf{r}\|^2 \tag{5.25}$$

$$= \lambda^{2nd} \|(\mathbf{S} - \mathbf{I})\hat{\mathbf{r}}\|^2 \tag{5.26}$$

because the $\hat{r}_i^*$ are a linear combination of the ranges of other nodes. $\mathbf{I}$ is the identity matrix, and the rows of $\mathbf{S}$ select and weight (based on image pixel colours) the 2nd-Order neighbourhood of each node in turn. The matrix has a number of diagonal bands.

The terms can be further expanded so that

$$\Psi = \lambda^{dat} \left[\hat{\mathbf{r}}^{\mathbf{T}}\mathbf{D}^{\mathbf{T}}\mathbf{D}\hat{\mathbf{r}} - 2\mathbf{z}^{\mathbf{T}}\mathbf{D}^{\mathbf{T}}\mathbf{D}\hat{\mathbf{r}} + \mathbf{z}^{\mathbf{T}}\mathbf{D}^{\mathbf{T}}\mathbf{D}\mathbf{z}\right] \tag{5.27}$$

$$\Phi^{1st} = \lambda^{1st} \left[\hat{\mathbf{r}}^{\mathbf{T}}\mathbf{F}^{\mathbf{T}}\mathbf{F}\hat{\mathbf{r}}\right] \tag{5.28}$$

$$\Phi^{2nd} = \lambda^{2nd} \left[\hat{\mathbf{r}}^{\mathbf{T}}(\mathbf{S} - \mathbf{I})^{\mathbf{T}}(\mathbf{S} - \mathbf{I})\hat{\mathbf{r}}\right] \tag{5.29}$$

If the Energy function that must be minimized is given by $E = \Psi + \Phi^{1st} + \Phi^{2st}$ then

we can simplify to

$$E = \hat{\mathbf{r}}^{\mathbf{T}}\mathbf{A}\hat{\mathbf{r}} - 2\mathbf{b}^{\mathbf{T}}\hat{\mathbf{r}} + \mathbf{z}^{\mathbf{T}}\mathbf{b} \tag{5.30}$$

$$\mathbf{A} = \lambda^{2nd}\left[(\mathbf{S}-\mathbf{I})^{\mathbf{T}}(\mathbf{S}-\mathbf{I})\right] + \lambda^{1st}\left[\mathbf{F}^{\mathbf{T}}\mathbf{F}\right] + \lambda^{dat}\left[\mathbf{D}^{\mathbf{T}}\mathbf{D}\right] \tag{5.31}$$

$$\mathbf{b} = \lambda^{dat}\left[\mathbf{D}^{\mathbf{T}}\mathbf{D}\mathbf{z}\right] \tag{5.32}$$

Taking the partial derivative of E with respect to $\hat{\mathbf{r}}$ gives,

$$\frac{\partial_E}{\partial_{\hat{\mathbf{r}}}} = (\mathbf{A}^T + \mathbf{A})\hat{\mathbf{r}} - 2\mathbf{b} \tag{5.33}$$

$$= 2\mathbf{A}\hat{\mathbf{r}} - 2\mathbf{b} \tag{5.34}$$

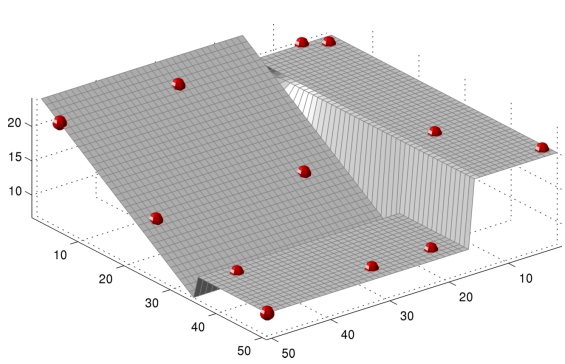
since $\mathbf{A}$ must be symmetric, and setting it to zero to find the $\hat{\mathbf{r}}$ which minimizes E yields the familiar form

$$\boxed{\mathbf{A}\hat{\mathbf{r}} = \mathbf{b}} \tag{5.35}$$

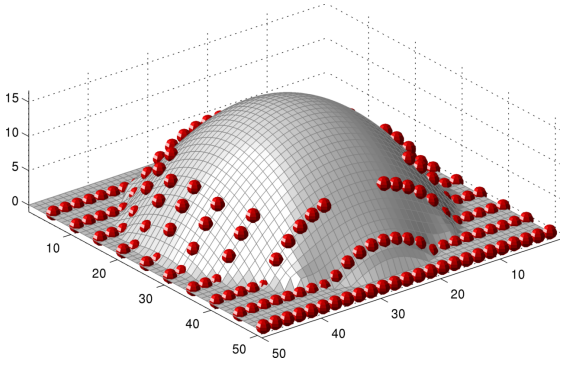
Sparse linear systems such as this may be solved quickly by a number of standard algorithms. We chose to use the 'backslash' operator in Matlab.

5.5 Synthetic Examples

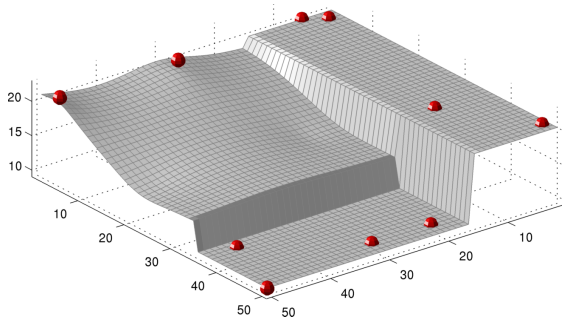
In order to demonstrate the power of our 2nd-Order smoothness term, we apply it to the two synthetic examples used earlier in this chapter. Figure 5.14 shows the results of varying the relative power of the 1st and 2nd-Order terms. Significantly, the 2nd-Order term is able to fully reconstruct the inclined plane in the 'Three Planes' dataset, and it makes a very reasonable estimate of the curved dome. This is remarkable given the sparsity of the known measurements. The generated curved surface is the smoothest surface that can explain the existing measurements and minimize the



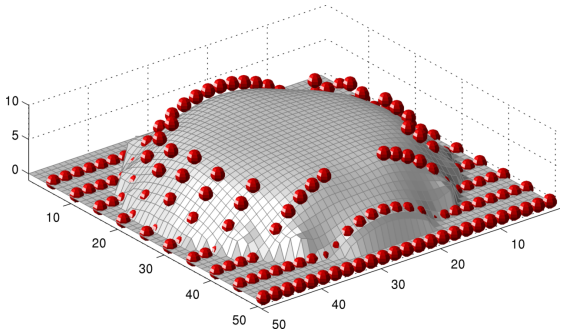
(a) Three Planes, All 2nd-Order
$\lambda^{dat} = 0.70, \lambda^{1st} = 0.00, \lambda^{2nd} = 60.00$



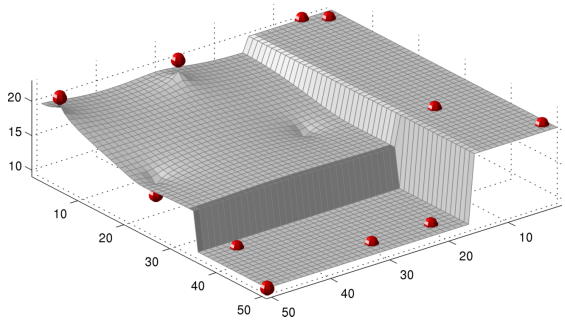
(b) Dome, All 2nd-Order
$\lambda^{dat} = 0.70, \lambda^{1st} = 0.00, \lambda^{2nd} = 60.00$



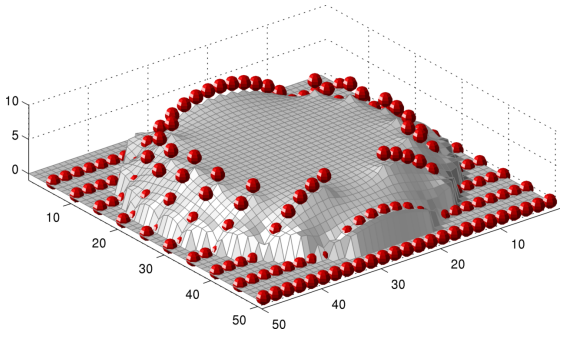
(c) Three Planes, Midway
$\lambda^{dat} = 0.70, \lambda^{1st} = 0.13, \lambda^{2nd} = 33.30$



(d) Dome, Midway
$\lambda^{dat} = 0.70, \lambda^{1st} = 0.13, \lambda^{2nd} = 33.30$



(e) Three Planes, All 1st-Order
$\lambda^{dat} = 0.70, \lambda^{1st} = 0.30, \lambda^{2nd} = 0.00$



(f) Dome, All 1st-Order
$\lambda^{dat} = 0.70, \lambda^{1st} = 0.30, \lambda^{2nd} = 0.00$

Figure 5.14: Results on the synthetic datasets, varying the balance between 1st-Order and 2nd-Order terms. In all cases, we set the colour variance term $\sigma_c = 1.0$ and the strength of the Data cost to be 0.3 times the sum of the Smoothness strengths. Our 2nd-Order smoothness term does an excellent job both of reproducing the inclined plane (top left), but also of inferring the curved surface of the Dome example (top right).

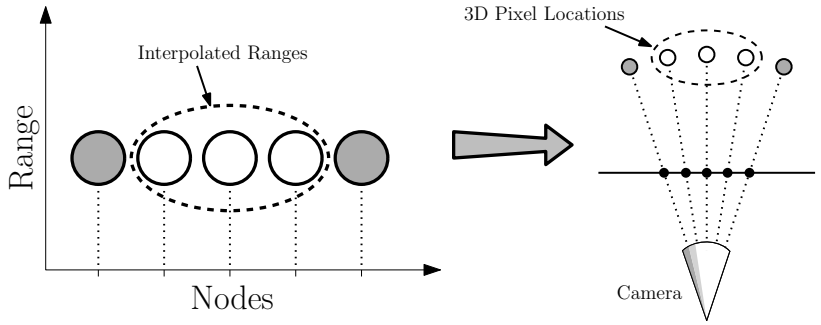


Figure 5.15: If the geometry of the camera model is not taken into account undesirable distortions arise in the reconstructed scene. Here the optimization using an affine model has produced a cluster of range nodes (via interpolation in this case) of equal range: they all lie on the same plane. Projecting the pixels attached to these nodes back into 3D results in a distorted curved surface.

violation of the 2nd-Order prior.

5.5.1 Projective Case

So far, all the methods we have discussed have made the implicit assumption that the camera rays through all the pixels are parallel - the camera has an affine projection model. Real cameras do not behave like this, so for completeness we now address the full projective case, and develop a model which can be incorporated into the exisitng linear MRF framework. Figure 5.15 shows the problem: Surfaces that are planar in the pixel-range representation are warped by the projection through the camera model into Cartesian coordinates. In some cases an affine model may be sufficient in which case our method will yield a direct solution for $\hat{\mathbf{r}}$ in one step. However a complete analysis of the problem demands we examine the more general case of a projective camera. For such a camera we need a way to handle the divergent nature of rays from the camera centre and yet retain the benefits of the problem being in a linear form. We do this by introducing an iterative scheme in which a sequence of solutions of $\hat{\mathbf{r}}$ are obtained with each iteration further minimizing the overall cost function.

Consider the interpolation case previously discussed as Equation 5.15. Figure 5.16

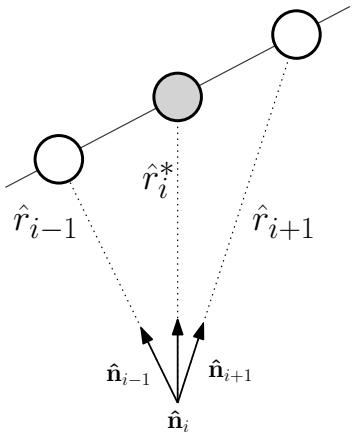


Figure 5.16: Interpolation in the projective case is more complicated than the affine case. It still uses a linear combination of the two neighbouring nodes, but the prediction for the central node is now of the form $\hat{r}_i^* = \alpha_i(\hat{\mathbf{n}}_i\hat{\mathbf{n}}_{i+1})r_{i+1} + (1-\alpha_i)(\hat{\mathbf{n}}_i\hat{\mathbf{n}}_{i-1})r_{i-1}$. This can be interpreted as projecting the two neighbour nodes onto the central ray before interpolating between them. The interpolation value α_i depends on the current 'best guess' of the node ranges.

shows the projective case. To minimize local curvature, the ith node must be placed such that it lies on a line drawn between its two neighbours. Unlike the affine case, the neighbours are not equal distances away, so we cannot simply use equal interpolation weights of 0.5. Calculating the range which puts the ith node on the line between its neighbours gives:

$$\hat{r}_i^{*(Interp)} = \alpha_i(\hat{\mathbf{n}}_i \cdot \hat{\mathbf{n}}_{i+1})r_{i+1} + (1-\alpha_i)(\hat{\mathbf{n}}_i \cdot \hat{\mathbf{n}}_{i-1})r_{i-1} \tag{5.36}$$

$$\alpha_i = \frac{|\hat{r}_{i-1}\hat{\mathbf{n}}_{i-1} \times \hat{\mathbf{n}}_i|}{|(\hat{r}_{i-1}\hat{\mathbf{n}}_{i-1} - \hat{r}_{i+1}\hat{\mathbf{n}}_{i+1}) \times \hat{\mathbf{n}}_i|} \tag{5.37}$$

where $\hat{\mathbf{n}}_i$ is the camera ray direction for the ith node. In effect, the two neighbouring nodes are being projected onto the central ray, and then the interpolation can happen in 1D.

Similar expressions can be derived for the left and right extrapolation:

$$\hat{r}_i^{*(LExtrap)} = \beta_i(\hat{\mathbf{n}}_{i-1} \cdot \hat{\mathbf{n}}_i)r_{i-1} + (1-\beta_i)(\hat{\mathbf{n}}_{i-2} \cdot \hat{\mathbf{n}}_i)r_{i-2} \tag{5.38}$$

$$\beta_i = \frac{|\hat{r}_{i-2}\hat{\mathbf{n}}_{i-2} \times \hat{\mathbf{n}}_i|}{|(\hat{r}_{i-1}\hat{\mathbf{n}}_{i-1} - \hat{r}_{i-2}\hat{\mathbf{n}}_{i-2}) \times \hat{\mathbf{n}}_i|} \tag{5.39}$$

and

$$\hat{r}_i^{*(RExtrap)} = \gamma_i(\hat{\mathbf{n}}_{i+2} \cdot \hat{\mathbf{n}}_i)r_{i+2} + (1-\gamma_i)(\hat{\mathbf{n}}_{i+1} \cdot \hat{\mathbf{n}}_i)r_{i+1} \tag{5.40}$$

$$\gamma_i = \frac{|\hat{r}_{i+2}\hat{\mathbf{n}}_{i+2} \times \hat{\mathbf{n}}_i|}{|(\hat{r}_{i+1}\hat{\mathbf{n}}_{i+1} - \hat{r}_{i+2}\hat{\mathbf{n}}_{i+2}) \times \hat{\mathbf{n}}_i|} \tag{5.41}$$

If we can assume that the interpolation coefficients α_i, β_i and γ_i are fixed, then the expressions for $\hat{r}_i^*$ are simply linear combinations of neighbouring node ranges, as before. The dot products involving the ray normals will be constant because the camera model does not change.

This suggests an iterative scheme, where we fix the γ values and find the ranges, then recompute the γ values and so on. Equation 5.35 becomes

$$\mathbf{A}_{m-1}\hat{\mathbf{r}}_m = \mathbf{b} \tag{5.42}$$

where m is the current timestep and

$$\mathbf{A}_{m-1} = \lambda\left[(\mathbf{S}_{m-1} - \mathbf{I})^T(\mathbf{S}_{m-1} - \mathbf{I}) + \mathbf{F}^T\mathbf{F} + k\mathbf{D}^T\mathbf{D}\right] \tag{5.43}$$

The matrix $\mathbf{S}_{m-1}$ (See Equation 5.26) is constructed by making the linearizing assumption that the iteration at time m will preserve the interpolation coefficients α_i, β_i and γ_i. Without this assumption the non linear nature of the camera model would make a linear form of the problem impossible. After each iteration step, the interpolation coefficients are recalculated based on the current range estimates, $\hat{\mathbf{r}}_m$.

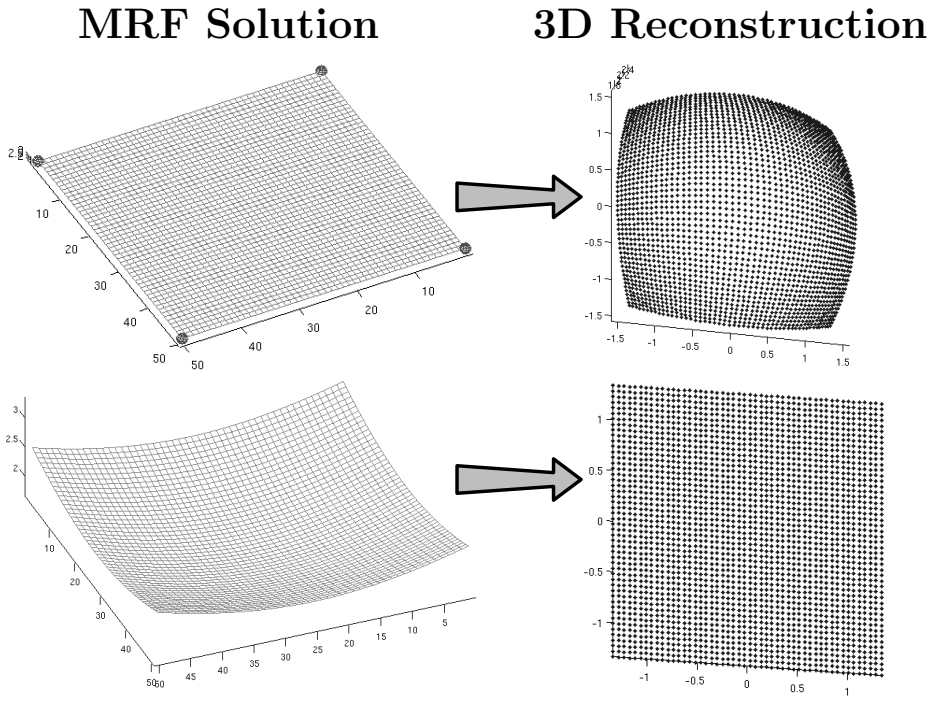


Figure 5.17: The MRF approach makes the assumption that the camera has an affine projection model (ie. the rays are all parallel). This is not the case for real cameras. When the pixel ranges are projected back out into 3D space, formerly flat surfaces come out curved. What is needed is a way to cause the MRF to warp its output, such that the 3D Reconstruction is correct.

It is reasonable at this point to query the gains made by incorporating the projective model over the simple affine case. Figure 5.17 shows the results of applying the iterative projective correction. It shows the warped surface created by the MRF, which when projected through the camera model becomes a flat surface. We offer no formal convergence guarantees, but empirically we find that the algorithm converges very quickly, usually requiring only around 2 iterations - the curvature due to ray divergence being small compared to the scale of the measurements.

5.6 Practical Considerations

When it comes to applying this approach to real data collected from a mobile robot, several practical issues arise. Firstly we must address the effects of so called *mixed measurements* (See Tuley et al. [111]) near range discontinuities. If the laser spot is large enough to fall across two surfaces, the returned range is a weighted sum of the two possible ranges and does not correspond to a measurement on any real surface, as shown in Figure 4.9. Secondly we must consider how to handle the consequences of an almost inevitably imperfect calibration between laser and camera coordinate frames.

5.6.1 A Laser Sensor Model

Equation 5.21 introduced a weighting term σ_i expressing confidence in a particular range measurement r_i. As mixed measurements occur near range discontinuities and we are using image appearance discontinuities as depth change cues, we propose the following form for a model of confidence in the laser measurements:

$$\sigma_i(\mathcal{I}) = 1 - \eta \exp\left(-\frac{\rho(i, \bigtriangledown\mathcal{I})^2}{2\sigma_l^2}\right) \tag{5.44}$$

where σ_i is the confidence in the i^{th} laser measurement, σ_l is a parameter controlling the radius of influence of the downweighting effect, $\eta \in [0, 1]$ is a scale factor[3]. affecting the strength of the downweighting and $\rho(i, \bigtriangledown\mathcal{I})$ is the pixel distance between the measured range node $\mathbf{R}_i$ and the nearest image edge. An advantage of this model is that it goes some way to mitigating issues arising from camera-laser calibration errors. It is at the range discontinuities that alignment errors have dramatic effects; misaligned step changes in range and appearance can cause marked rippling effects in the solution surface. By diluting the confidence in all laser points lying near image edges we simultaneously account for mixed measurements and small calibration errors.

5.7 Results

We now turn to processing some real data. We used a SICK LMS200 laser scanner mounted on a robot to capture laser data as it was actuated in a nodding fashion. Images were captured by a camera mounted above the laser with a wide angle lens. The image used in this case was 518 by 259 pixels resulting in some 134,162 range nodes and is shown in Figure 5.18 with laser measurements projected into it. The reconstructed model is shown alongside, which was produced with parameters: $\sigma_c = 100, \lambda^{dat} = 0.5, \lambda^{1st} = 0.15, \lambda^{2nd} = 70$. Using second-order smoothness alone provides reasonable results, but tends to introduce 'rippling' artifacts around noisy measurements. A small amount of first-order smoothness is necessary to damp the oscillations. We show an outdoor result of the same problem size and using the *same* tuning parameters in Figure 5.19.

We now present some numerical analysis of the performance of our approach. It is a hard task to obtain a ground truth geometry for the complete real scene. Instead of

[3]In this work, the parameters were set empirically. We chose $\eta = 0.1$, to almost completely ignore laser measurements on image discontinuities, with a small $\sigma_l = 1$ pixel to limit the radius of influence.

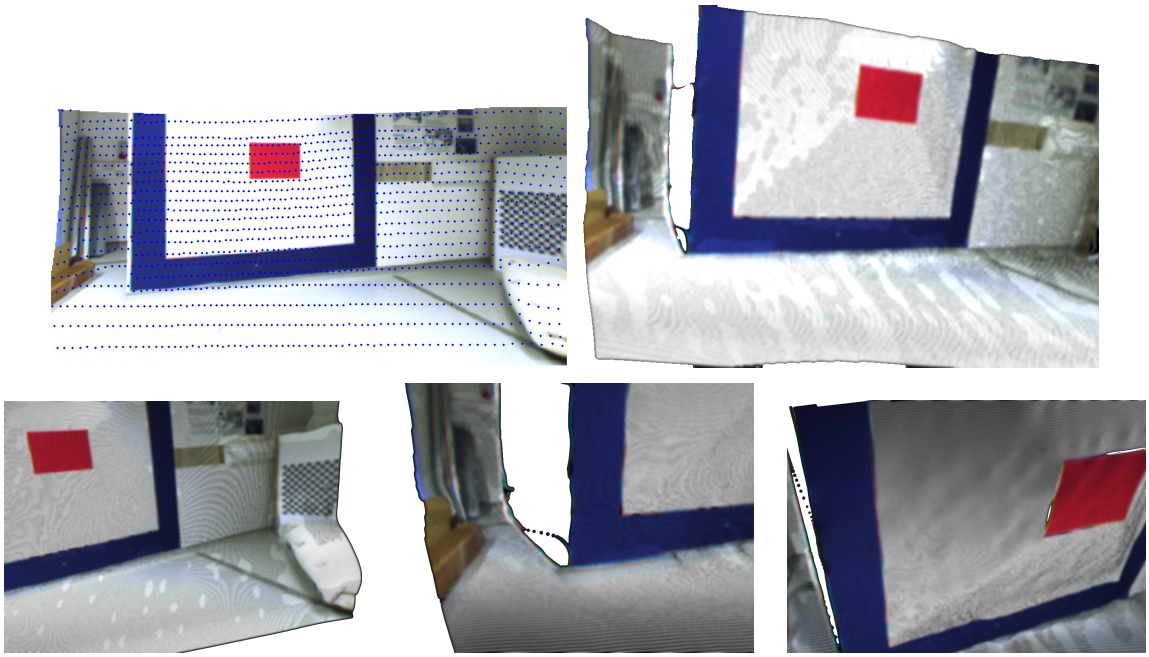


Figure 5.18: Results from an indoor dataset. The original image and laser measurements are shown in the top left, with a series close-ups of the reconstructed model following. Note the detail of the smooth floor and inferred sharp range discontinuity between two walls. The parameters used to produce this reconstruction were: $\sigma_c = 100, \lambda^{dat} = 0.5, \lambda^{1st} = 0.15, \lambda^{2nd} = 70$

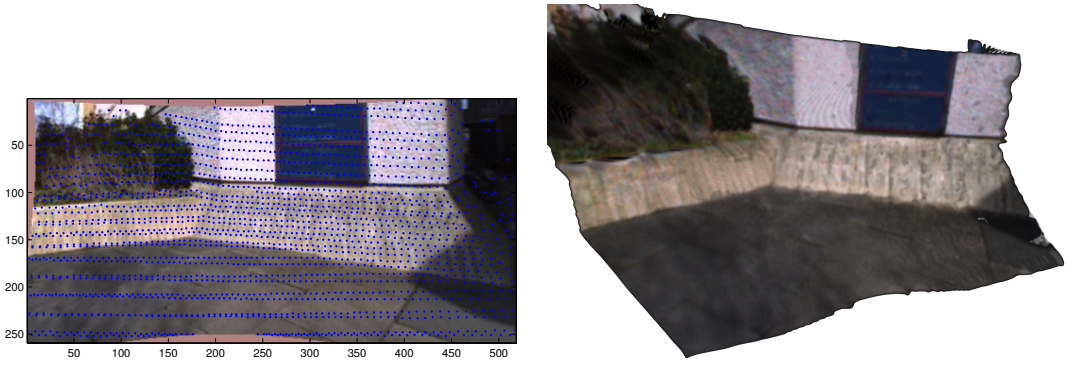
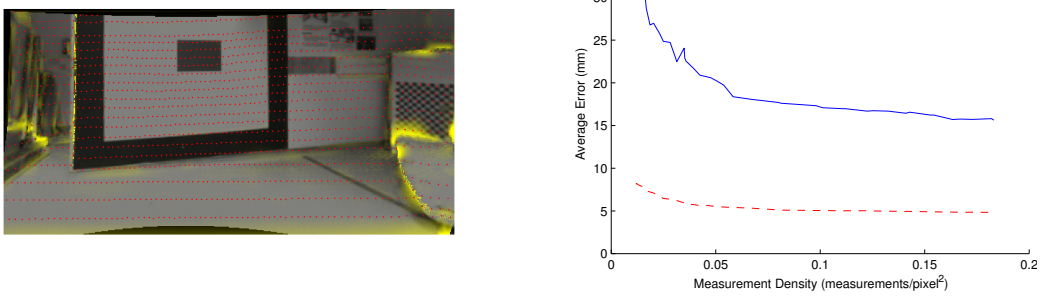


Figure 5.19: Results from an outdoor dataset. On the left is the image with laser measurements overlaid. On the right is the reconstructed model.The parameters used to produce this reconstruction were: $\sigma_c = 100, \lambda^{dat} = 0.5, \lambda^{1st} = 0.15, \lambda^{2nd} = 70$



(a) **(b)**

Figure 5.20: The left image shows a comparison of range estimates to ground truth laser data for the indoor case. Areas in yellow show deviation from ground truth, with higher intensity representing larger errors. Laser measurements are shown in red. The graph shows average error of the estimate relative to the mean density of range measurements, when compared to hold out set laser measurements. The laser has a typical measurement accuracy of ±15mm.

comparing pixel ranges to ground truth we compare them to laser measurements taken of the scene over a long period of time and which are not used in the optimization. Concretely, we collect a very dense cloud of laser data at the scene and draw from that a small sparse test set with which we reconstruct the scene shown in Figure 5.18. The remaining laser data constitutes a dense hold out set. For each unused laser measurement we can compare measured range to estimated range. Figure 5.20a shows regions of the workspace which contain pixels with significant errors. These tend to correspond to areas having large range discontinuity but relatively weak image edges.

It is also instructive to consider how the accuracy of our approach depends on the density of laser measurements. Figure 5.20b shows how the statistics (mean and median) of the pixel range errors change as a function of measurement density. Note that as expected, as measurement density increases the accuracy tends to that of the laser itself (around ±15mm). The results given in Figures 5.18 and 5.19 are operating in the 0.01 measurements/pixel^2 region.

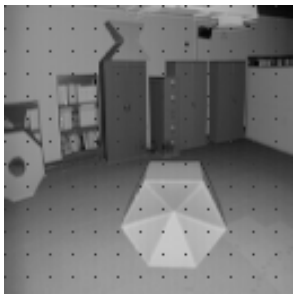
5.8 Comparison With Other Methods

To give a fair comparison with the other methods described in Section 5.3 we used a standard, readily available dataset. The 'USF Range Image Database' [4] is hosted by the CESAR Lab at Oak Ridge National Laboratory, TN, USA. It consists of a number of indoor range and reflectance images taken with an Odetics laser range finder. It has the advantage over our own data of having range measurements for every pixel in the image, which is much more convenient for measuring error statistics. The reflectance images are monochrome, and the range measurement resolution is 0.03m.

First, we 'trained' each of the methods on the 'engine.0' range scan from the database (Figure 5.21), with a measurement density of 0.01meas/pixel^2. The training was carried out by repeatedly running each method on the example and finding the tuning parameters that minimized the Mean Squared Error over all range nodes. Essentially we performed a grid search over the parameter space for each method. In some cases the parameter space had too many dimensions for this to be tractable, so some parameters were chosen heuristically. Crucially, once the parameters were tuned, they were *untouched* for the rest of the experiment, and we did not use the 'engine.0' dataset to produce any results. The parameters used for all of the experiments are summarized in Figure 5.21.

We ran each of the algorithms on three randomly chosen USF datasets, with three different measured node densities. As well as the interpolation methods previously discussed, we implemented a simple Bilinear Interpolation method to provide a baseline of performance. In the results tables, it is referred to as 'BILINEAR'. The results of our experiments are shown in Tables 5.1, 5.2 and 5.3. The errors are measured in metres; for each experiment we give the Mean Squared Errors (MSE), the Median Error and the Variance of the error over the whole image.

[4] http://marathon.csee.usf.edu/range/DataBase.html



(a) 'engine.0' image

Method	σ_c	σ_p	λ^{dat}	λ^{1st}	λ^{2nd}
BILINEAR	-	-	-	-	-
NR	-	-	-	-	-
NRC	0.401	0.966	-	-	-
MLI	-	-	-	-	-
LIC	0.306	-	-	-	-
PLIC	-	-	-	-	-
MRF 1st-Order	22.360	-	0.700	0.300	-
MRF 2nd-Order	22.360	-	0.700	0.075	45.000

Figure 5.21: Parameter settings used for the different interpolation methods. A dash indicates that the method does not use a particular parameter. The values were obtained by training on the 'engine.0' range scan from the USF Range Image Database. The known measurements used in training are shown as dots on the image.

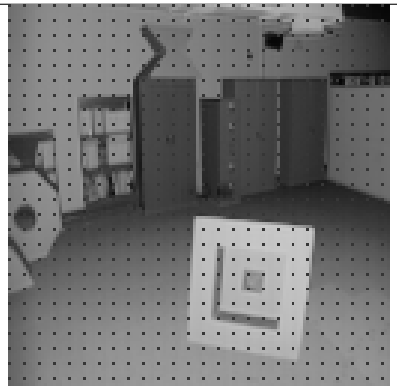
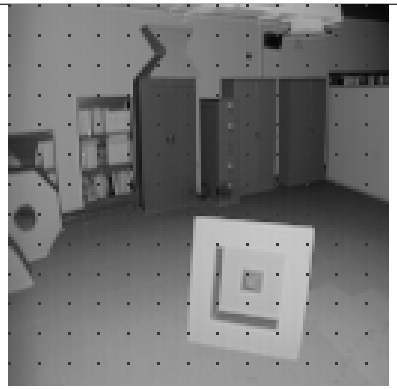
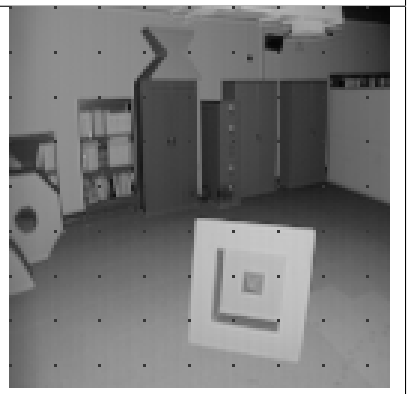
‘Ashtray.0’ Range Scan									
Resolution (meas/pixel^2)	0.040			0.010			0.004		
									
	MSE	Median	Var	MSE	Median	Var	MSE	Median	Var
Bilinear Interpolation	0.0617	0.0220	0.0508	0.1489	0.0512	0.1131	0.2291	0.1147	0.1529
NR (Nearest Nbs)	0.0992	0.0366	0.0836	0.2480	0.1098	0.1959	0.3677	0.1464	0.2646
NRC (Nearest Nbs + Colour)	0.1517	0.0732	0.1237	0.3373	0.1098	0.2578	0.3918	0.1830	0.2691
MLI (NatNbs)	0.0606	0.0244	0.0495	0.1588	0.0769	0.1181	0.2063	0.1076	0.1381
LIC (NatNbs + Colour)	0.0500	0.0220	0.0425	0.1013	0.0656	0.0766	0.1110	0.0803	0.0772
PLIC	0.0834	0.0366	0.0715	0.1489	0.0764	0.1164	0.1660	0.1098	0.1149
MRF (1st-Order Only)	0.0527	0.0398	0.0422	0.0976	0.1146	**0.0607**	0.1513	0.1786	0.0830
MRF (1st & 2nd-Order)	**0.0481**	**0.0192**	**0.0420**	**0.0828**	**0.0474**	0.0647	**0.1090**	**0.0735**	**0.0771**

Table 5.1: Results for each of the implemented methods on the ‘Ashtray.0’ range scan. Units are metres.

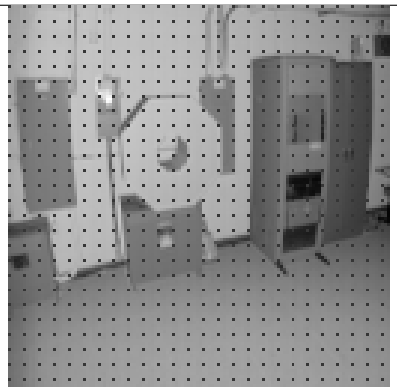
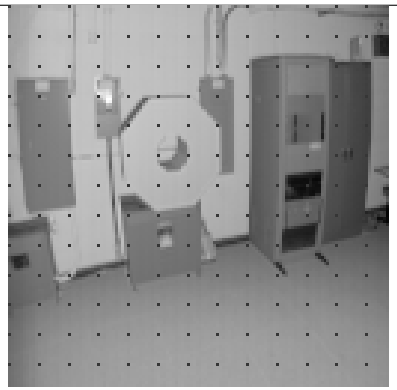
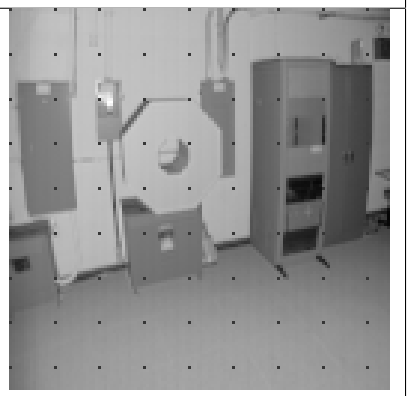
‘r1.24’ Range Scan									
Resolution (meas/pixel^2)	0.040			0.010			0.004		
									
	MSE	Median	Var	MSE	Median	Var	MSE	Median	Var
Bilinear Interpolation	0.0223	**0.0146**	0.0185	0.0391	0.0366	0.0285	0.0454	**0.0512**	0.0317
NR (Nearest Nbs)	0.0315	0.0366	0.0254	0.0718	0.0732	0.0509	0.0786	0.1098	0.0498
NRC (Nearest Nbs + Colour)	0.0545	0.0366	0.0430	0.1231	0.0732	0.0870	0.1194	0.1098	0.0784
MLI (NatNbs)	0.0219	0.0195	0.0180	0.0466	0.0513	0.0331	0.0572	0.0547	0.0406
LIC (NatNbs + Colour)	0.0191	0.0171	0.0160	0.0432	0.0428	0.0317	0.0567	0.0480	0.0416
PLIC	0.0242	0.0327	0.0196	0.0491	0.0624	0.0345	0.0637	0.0741	0.0423
MRF (1st-Order Only)	0.0144	0.0300	0.0104	0.0400	0.0867	0.0232	0.0614	0.1214	0.0328
MRF (1st & 2nd-Order)	**0.0113**	0.0174	**0.0089**	**0.0267**	**0.0361**	**0.0191**	**0.0410**	0.0579	**0.0275**

Table 5.2: Results for each of the implemented methods on the ‘r1.24’ range scan. Units are metres.

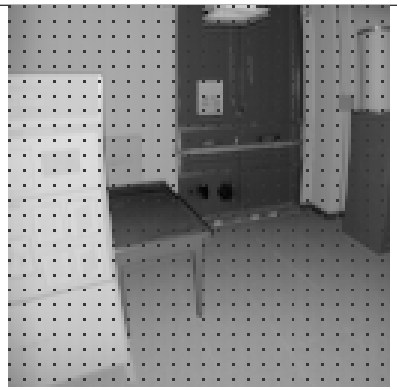
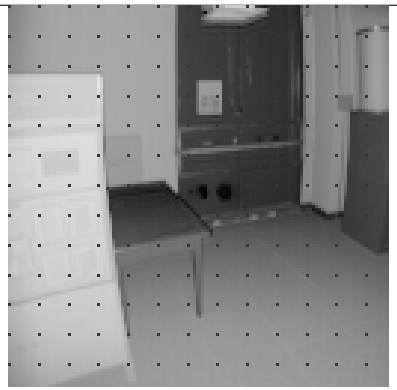
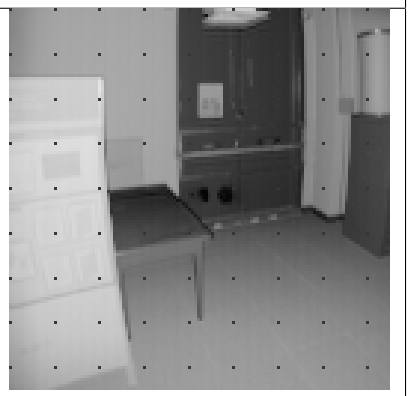
‘r0.28’ Range Scan									
Resolution (meas/pixel^2)	0.040			0.010			0.004		
									
	MSE	Median	Var	MSE	Median	Var	MSE	Median	Var
Bilinear Interpolation	0.0479	**0.0146**	0.0423	0.0922	**0.0293**	0.0755	0.1286	0.0415	0.0964
NR (Nearest Nbs)	0.0646	0.0366	0.0575	0.1618	0.0732	0.1372	0.1989	0.0732	0.1609
NRC (Nearest Nbs + Colour)	0.1216	0.0366	0.1047	0.2966	0.0732	0.2414	0.2982	0.0732	0.2338
MLI (NatNbs)	0.0453	0.0171	0.0398	0.0939	0.0474	0.0742	0.1031	0.0508	0.0770
LIC (NatNbs + Colour)	0.0363	0.0159	0.0325	0.0638	0.0404	0.0516	0.0820	**0.0410**	0.0646
PLIC	0.0591	0.0311	0.0525	0.1319	0.0477	0.1116	0.1520	0.0665	0.1233
MRF (1st-Order Only)	0.0297	0.0260	0.0253	0.0477	0.0660	0.0328	**0.0690**	0.0903	**0.0452**
MRF (1st & 2nd-Order)	**0.0253**	0.0166	**0.0226**	**0.0352**	0.0315	**0.0287**	0.0966	0.0454	0.0828

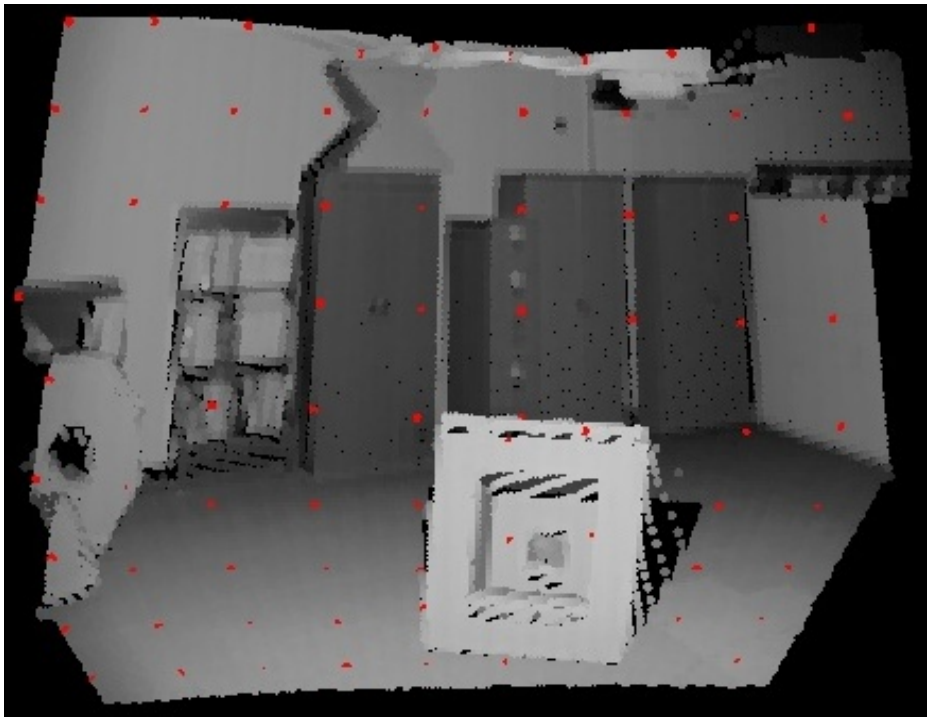
Table 5.3: Results for each of the implemented methods on the ‘r0.28’ range scan. Units are metres.

The MRF with 2nd-Order smoothness term compares very favourably with our implementations of the other methods. In almost all cases it has the lowest Mean Squared Error scores and smallest error variance. The cases were it does not perform so well generally correspond to surfaces with small area, and therefore have very few samples lying on them. In these cases the MRF has large latitude to infer an incorrect surface, whereas the greedy interpolation schemes must have their estimates bounded by the ranges of nearby measurements - no extrapolation can ever occur. Nonetheless, the MRF with 2nd-Order smoothness still performs well with sparse measurements. Figure 5.22 shows the resulting 3D surface for the most sparse experiment on the 'ashtray.0' range scan.

5.9 Chapter Summary

We have introduced a novel technique for fusing sparse laser data and images to enable a dense 3D scene reconstruction. Above and beyond existing prior work this technique uses a second order smoothness term which allows it to extrapolate both planar and curved surfaces. The problem is formulated as the solution of a sparse linear system, which allows the use of fast optimization techniques. In particular, we have shown that a judicious framing of the affine case leads to a linear problem. Furthermore, we have described an iterative scheme which allows a more realistic projective camera model to be folded into the overall framework. In order to account for step changes in range commonly found in real environments, a laser sensor model was developed that models the likelihood of mixed measurements commonly encountered in such situations. The technique was applied to both illustrative synthetic cases as well as real data recorded in indoor and outdoor scenes containing challenging geometry.

One might ask why we should stop at 2nd order smoothness terms. We declined to consider higher order curvature terms due to issues of computational complexity.



(a) Ground truth

(b) MRF with 2nd-Order Smoothness

Figure 5.22: 3D rendering of the range image for the 'ashtray.0' range scan. Even with a particularly sparse measurement density of 0.0044 meas/pixel^2, the MRF with 2nd-Order smoothness is able to infer the unknown ranges with remarkable fidelity.

In particular, the clique size would increase greatly, and the linear MRF framework would not support the curvature models required.

Our results showed that with well-chosen tuning parameters, our method significantly out performed a number of contemporary algorithms. Nevertheless, there is room for improvement. In particular we must consider how we can increase robustness to erroneous laser measurements (away from image edges) and how we might fuse multiple scenes in a principled way. The flip side of this problem is handling bona-fide discontinuities in range when there is no change in image appearance and vice versa.

This point marks the end of the direct exploitation of range data and the beginning of a focus on high-precision timing. The motivation for this transition lies in the observation that the intrinsic value of time series measurements is limited by the precision and our confidence in the data's time stamps. Better timestamping is synonymous with better data. Improved timestamping yields improved data.

Chapter 6

Clock Synchronization

> "The only reason for time is so that everything doesn't happen at once."
>
> *Albert Einstein*

6.1 Introduction

We now move on to describe work done to provide highly accurate time synchronization between networked computers. We shall describe a system implemented for the Mission Oriented Operating Suite (MOOS) [77] - robot middleware software for communication between distributed processes. The techniques are however applicable to clock synchronization in general, which is of vital importance in networking and communication technologies.

The architecture of MOOS allows communication between multiple processes running on many different machines on a network. A fundamental requirement of MOOS is that processes which provide time-stamped data have synchronized clocks. This ensures that time stamps generated by one process are consistent with those of all other processes.

A simple example would be that of a robot with the odometry system interfaced to one physical computer, and sensors interfaced to another physical computer. In

order to create a map from the sensor data, it is necessary to know exactly where the robot was when an observation was taken. Given a timestamp for the observation, we may query the odometry buffer to obtain the pose of the robot for that timestamp. A discrepancy between the clocks of the two computers would cause an incorrect pose to be returned.

A common approach to clock synchronization in a robotics context is to run an NTP (Network Time Protocol) [70] server on one computer and have the others adjust their clocks to it [59]. The clocks are brought slowly into alignment by varying their frequencies by small amounts. This is undesirable in a robotics context because it can take many hours to synchronize clocks via NTP, and the frequency changes can cause inconsistencies in timestamps. It also requires manual configuration of all the computers, independent of the middleware system. An incorrect configuration (ie. synchronizing to the wrong server) could be disastrous and difficult to diagnose.

A variant of NTP, known as SNTP (Simple Network Time Protocol) which is the default for synchronization in Microsoft Windows and some Linux distributions can cause particular problems. Synchronization is performed periodically and involves applying an offset to the client clock to immediately bring it in line with the server. The greater the disparity in clock frequencies, the larger the offset jumps will be. In some cases this can result in time appearing to go backwards.

What is needed is a system which operates transparently to the user, capable of rapidly synchronizing clocks and able to adjust to clock upsets such as frequency changes and instantaneous resets (offset jumps) which may be exhibited by clocks under the control of NTP or SNTP. In other words, we require a system which maintains synchronization *despite* the use of NTP.

We will first frame the problem and then discuss existing approaches to a solution. We will examine in depth two approaches proposed by other authors, and then describe our own real-time network based clock synchronization algorithm which is

capable of synchronizing clocks to arbitrary skew accuracy if given enough time. We call our algorithm the *Timestamp-based Incremental Clock Synchronization Scheme*, or *TICSync* for short. Our particular contributions are: An efficient incremental algorithm, a method for deriving probabilistic convergence properties, and a method for online detection of clock upsets, known as TICSync+.

6.2 How bad can clocks be?

Modern PCs use two methods for keeping track of time. On the motherboard is the *hardware clock*, which has a quartz oscillator and a dedicated battery, so that it may keep time while the computer is turned off. These devices are low cost and temperature sensitive. Under normal conditions they should be expected to lose or gain up to 15 seconds per day.

The other method is the *software clock* (aka. *kernel clock*), which produces time interrupts used by the operating system for scheduling and time keeping. The software clock only runs while the computer is turned on, and is synchronized to the hardware clock at boot-up. Most Operating Systems have internal multipliers that allow the rate of the software clock to be adjusted. Programs such as the linux utility `adjtimex` attempt to set the rate to match that of the hardware clock.

Given the inaccuracies inherent in a regular PC's clock infrastructure, the software clocks on two computers cannot be expected to be without offset, or even running at the same speed. Between computers of identical hardware specification and software setup, we have observed clock skews of up to 170 ms per hour. In the context of an experiment gathering time-stamped data, this can cause significant data corruption.

6.3 Terminology and Assumptions

We now define the standard terminology that will be used throughout this chapter. The nomenclature was first introduced by Mills [70]. A *clock* is defined as a piecewise continuous monotonic function that may be differentiated twice everywhere except at a finite number of points where it may be reset. A *perfect* clock runs at a constant rate and always gives the 'true' time, $C(t) = t$. A clock is *correct* at some time instant t_i if $C(t_i) = t_i$ and *accurate* if $C'(t_i) = 1$. A clock is *stable* at some instant if it satisfies $C''(t_i) = 0$. So a *perfect* clock is correct, accurate and stable at all times.

For comparing two clocks, we will adopt the standard definitions from the literature:

Offset The difference in reported time between two clocks. The offset of clock a relative to clock b at time t is defined as $C_a(t) - C_b(t)$

Frequency The rate, $C'(t)$ at which a clock runs. This will always be a positive quantity (enforcing that time does not run backwards), except at a finite number of reset points.

Skew The difference in frequency of two clocks. The skew of clock a relative to clock b at time t is $C'_a(t) - C_b(t)$

Drift The drift of clock a relative to clock b at time t is defined as $C'''_a(t) - C'''_b(t)$

Consistency Consistent clocks share the same frequency, but may have some constant offset between their reported times: $C_a(t) = C_b(t) + c$

Computer clocks have limited resolution and may be seen as a piecewise continuous step function with step height equal to the resolution of the counter. In the following analysis, we shall treat the clock as a device which samples a continuous time function at discrete intervals.

We further assume that the clock resolution is smaller than the message transmission delays between computers. Those delays are modelled as a random variable, with each transmission delay being independent of adjacent delays.

6.4 Synchronization approaches

In the context of robotic middleware, there are many approaches to the problem of ensuring that separate devices produce consistent time stamps:

6.4.1 Clock adjustment techniques

Single shot The clocks of the computers are synchronized on a single occasion, before the mission begins. The offset is corrected, but skew is not addressed. This is successful only if the clocks are running at the same rate. If not, they will diverge over time.

Active synchronization Techniques such as NTP (Network Time Protocol) [70] bring clocks into alignment gradually over time by adjusting the rate of the client clock. The process can take many hours, since NTP tries to avoid large adjustments to the clock. If the clocks are fully synchronized before the mission starts, then time stamps can be accurate down to on the order of 200μs, but an incomplete synchronization will result in incorrect time stamps until convergence. Running NTP may not be desirable in some cases, especially as it is independent of the Middleware system and must be manually configured to synchronize to the correct server.

Occasional synchronization SNTP (Simple Network Time Protocol), as implemented in the Microsoft Windows Operating System, simply performs Single Shot updates at regular intervals (typically every few minutes). It does not

attempt to learn or adjust the frequency of the client clock. For clocks with large relative skew this can lead to unacceptable clock jumps.

6.4.2 Offset measurement techniques

These attempt to learn a mapping from client times to server times, which may be used to produce time stamps consistent with those at the server.

Single Shot The clock offset between client and server processes is measured once when the communications link is established and then assumed to be constant. The estimate will become progressively worse (depending on the skew) and may soon lead to data corruption.

Skew estimation The clock divergence between client and server processes is continually measured, and the skew estimated. This may be handled automatically and transparently to the user, since it does not rely on third party tools such as NTP.

The technique we describe here is of the latter type; we aim to *learn* the mapping between between two clock functions, so that knowing the time reported by one clock allows the recovery of the (estimated) time reported at the other clock at the same instant.

6.5 Message Timing Mechanism

Consider two computers communicating over a network: a client and a server. The client prepares a data packet, timestamps it with the current local (client) time and sends it to the server. If the clocks of the the two computers are perfectly synchronized then the server will observe a difference between the time of transmission and the time of receipt equal to the network propagation delay. If the clocks are consistent (equal

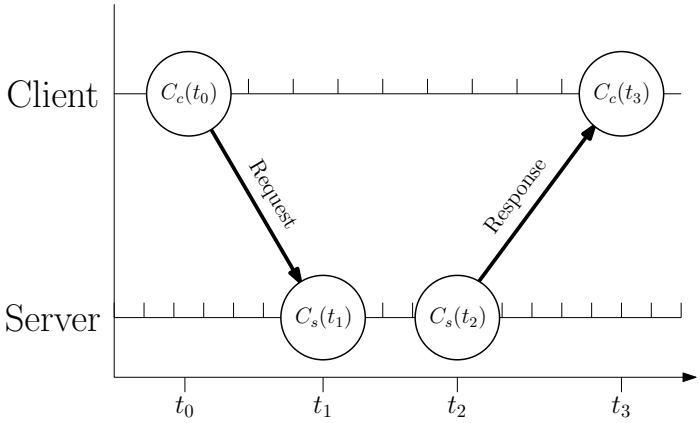


Figure 6.1: Timestamping mechanism between two computers. The client and server clocks are running at different frequencies, represented by the tick marks. When either computer sends or receives a packet, it adds a local timestamp. Possession of all four local timestamps allows a bounded offset estimate to be obtained. Note that propagation delays may be asymmetric.

frequency but have a constant offset) then the server will observe a difference equal to the sum of the offset and the network propagation delay. Crucially, the server is *unable* to recover the offset between the two clocks without having knowledge of the network delay.

We denote the server clock time as $C_s(t)$ and the client clock time as $C_c(t)$. It is convenient to assume that the server clock is perfect, so that $C_s(t) = t$. We may do this without loss of generality. We denote the true offset between the two clocks as $\tau(t)$. The offset may be positive or negative and is expressed as

$$\tau(t) = C_c(t) - C_s(t) \tag{6.1}$$

$$= C_c(t) - t \tag{6.2}$$

The offset may be estimated during the normal communications between the client and server. Consider Figure 6.1 which shows a single packet communication in each direction with local timestamps being stored at each *send* or *receive* event. We will refer to a packet from the client to the server as a *request* and a packet from the server to the client as a *response*.

The delays between the transmission and receipt events for each packet may be considered as random variabes, with distribution depending on network load, trip length, and CPU load at the client and server. It can not even be assumed that the delays are symmetric. Indeed, if the server is busy, it may not process and timestamp the request packets immediately, meaning that the outbound journey often takes longer than the return journey.

We now show that the difference between timestamps on the request packet gives a lower bound on the true offset at the time of receipt, and that the difference between timestamps on the response packet gives an upper bound on the true offset at the time of transmission. Because it will always take a finite positive amount of time for messages to travel through the system, we can state that,

$$t_0 < t_1 < t_2 < t_3 \tag{6.3}$$

We now show that we may obtain a *lower* bound on the true offset at t_1, $\tau(t_1)$ as,

$$\tau_{LB}(t_1) = C_c(t_0) - C_s(t_1) \tag{6.4}$$

Clocks always run forwards so $C_c(t_0) < C_c(t_1)$ and

$$\tau_{LB}(t_1) < C_c(t_1) - C_s(t_1) \tag{6.5}$$

$$= \tau(t_1) \tag{6.6}$$

A similar argument may be used to show that an *upper* bound on the true offset is

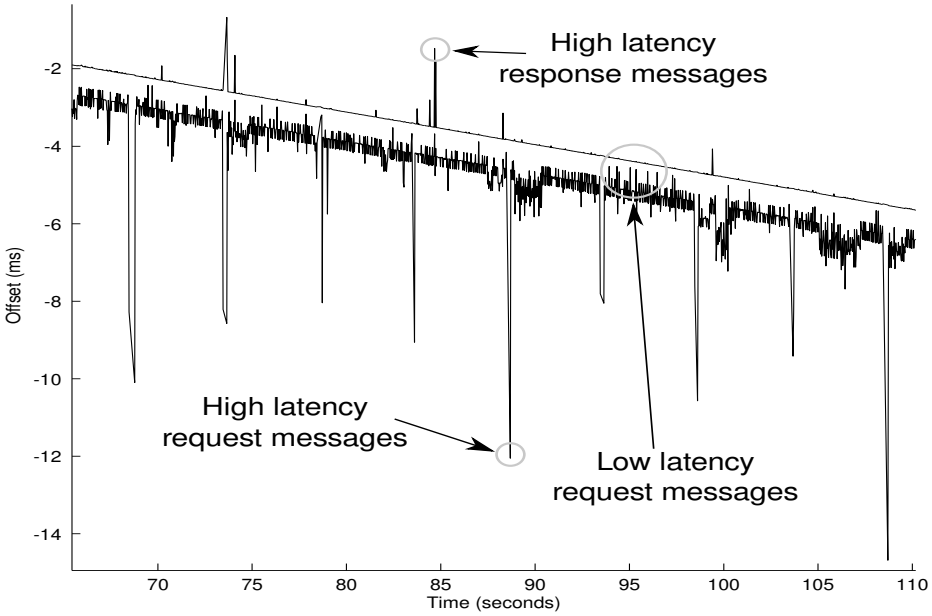


Figure 6.2: Typical offset measurements for a moderately loaded system running MOOS. The bottom line shows offsets measured from the request packet, and the bottom line shows offsets measured from the response packet. The two lines bound the true offset. Notice that request packets typically have higher latency than responses, yet there are still a few low latency request packets. This is a particular characteristic of the MOOS middleware and relates to the fact that request packets tend to contain more data than response packets. Also, if the server is busy it may not deal immediately with a request.

given by,

$$\tau_{UB}(t_2) = C_c(t_3) - C_s(t_2) \tag{6.7}$$

$$> C_c(t_2) - C_s(t_2) \tag{6.8}$$

$$= \tau(t_2) \tag{6.9}$$

6.6 The Bounds Corridor

Figure 6.2 shows typical upper and lower bound offset values measured by the client with a moderately loaded network. It can be seen that the upper and lower bound lines form a corridor in which the true offset value must lie. At a particular instant, the total Round Trip Time (RTT) is given by the difference between the upper bound and

the lower bound. As network load increases, packet journey times will increase, and the bounds will diverge, corresponding to larger RTTs. On a completely unloaded system, the RTT will reach a minimum value, 2δ, corresponding to the shortest possible time for an exchange between the client and server. The upper and lower lines forming the corridor will *never* intersect each other:

$$\tau_{UB} - \tau_{LB} \geq 2\delta \tag{6.10}$$

When there is a lot of traffic passing through the server, or the CPU is under high load, it may take longer than usual for the server to get round to processing a client request message. This explains the fact that in the example figure the lower bound line is noisier than the upper bound line. It can also be seen that occasionally a low latency message will slip through. We will show how these low latency messages can provide useful information, even if the total round trip time is large.

6.7 Clock skew model

Over periods on the order of tens of minutes, the effect of drift between clocks is typically small. It is reasonable to assume a constant-skew model:

$$C_c(t) = mt + c \tag{6.11}$$

where m and c are scalar parameters that must be learned by the client.

Using the definition for offset (equation 6.2) and the linear model (equation 6.11) we find that the offset at any time is given by

$$\tau(t) = C_c(t) - C_s(t) \tag{6.12}$$

$$= (m-1)t + c \tag{6.13}$$

The lower bound becomes

$$\tau_{LB}(t_1) = C_c(t_0) - C_s(t_1) \tag{6.14}$$

$$= mt_0 + c - t_1 \tag{6.15}$$

$$= \tau(t_1) - m(t_1 - t_0) \tag{6.16}$$

Similarly the upper bound becomes

$$\tau_{UB}(t_2) = C_c(t_3) - C_s(t_2) \tag{6.17}$$

$$= mt_3 + c - t_2 \tag{6.18}$$

$$= \tau(t_2) + m(t_3 - t_2) \tag{6.19}$$

If the client is able to learn the correct values of m and c over (ideally a relatively short) time, then given a local client timestamp, $C_c(t)$, the time at the server may be recovered trivially using

$$t = \frac{C_c(t) - c}{m} \tag{6.20}$$

From this point onwards we shall assume that the relative clock offset takes the form

$$\tau(t) = \alpha t + \beta \tag{6.21}$$

and that α and β are the quantities to be estimated.

6.8 Previous Work

The problem of synchronizing two clocks comes down to using the bounds corridor to estimate the skew and offset between the clocks, though many approaches consider only messages sent in one direction, For those algorithms the offset estimate will always be biased by the minimum network delay, δ.

6.8.1 Point estimate approaches

Early techniques based on exchanging timestamped messages made clock adjustments based on information from only the most recent exchange and would not estimate skew. The seminal work by Lamport [60] concentrated on the importance of determining the order of message events, rather than recovering exact time stamps. The algorithm assumes knowledge of the minimum network delay δ and requires that the clock of a process advances by at least one tick between any pair of message events (ie *send* or *receive*). Whenever a process receives a message that appears to have been sent after it arrived, the receiver's clock is set to be one tick greater than the transmit time plus the minimum propagation delay, δ. This ensures correct event ordering, but is not satisfactory for measurement applications. Later work by Lamport and Melliar-Smith [61] investigated the problem of achieving instantaneous synchronization given differing time reports from multiple processes, including an ability to deal with some processes reporting incorrect timing information.

Cristian's algorithm [28] refined the technique by considering bi-directional message passing, where Round Trip Times (RTT) can be measured. If a request-response pair has a RTT less than some threshold, then the requestor's clock is set to the response transmission time plus half the RTT. The accuracy of the technique is bounded by half the Round Trip Time. The technique is probabilistic in that the threshold should be set to ensure a good balance between the achieved accuracy and the number

of messages actually satisfying the threshold (ie. expected update interval). Gusella and Zatti [43] propose a similar algorithm to that of Cristian, assuming stable and accurate client clocks, and then show how it may be used to synchronize an ensemble of processes.

Lundelius and Lynch [65, 64] take measurements from multiple clocks and average them. They derive a lower bound on the achievable accuracy for a system of n clocks. Marzullo [67] describes the implementation of a time server in a distributed system. By assuming a known upper bound on clock skew, clocks in the system maintain an upper bound on their possible error which is updated whenever they are adjusted. The time server serves the value of the clock with the lowest error. Again, the accuracy of this algorithm is bounded below by the minimum message delay.

6.8.2 Filtering approaches

We now address approaches which use temporal filtering to obtain a more accurate result than is available from point estimates. Veitch et al. [114] use highly accurate driver-level time-stamping to minimize packet timing error. Within a sliding window, packet pairs with small Round Trip Times (RTTs) are selected and combined in a weighted average, with weights inversely proportional to RTT. Aweya et al. [6, 7] discuss a hardware based approach to skew estimation (they do not recover offset) using a jitter buffer and Phase Locked Loop. They suggest a linear regression approach to finding the skew. Recognizing the need to minimize stored data motivates their decision to use a sliding window and exponential smoothing of the estimate.

Paxson [89] takes an off-line multi-step filtering approach using bi-directional timestamped messages. First, the clock offset for each measurement pair is found by averaging the request and response delays (ie. taking a point mid-way between the two bounds). These values are then 'denoised' by splitting the signal into equal length segments and choosing the message pair with the smallest average delay in

each segment. If there are N measurements then the segment lengths are chosen to be the smaller of $\sqrt{N}$ samples or $(t_{N-1} - t_0)/\sqrt{N}$ seconds. Clock skew detection is performed by looking for downward trends in the data (or upward trends by mirroring the data and looking for downward trends). These are found by sweeping through the data and counting the number of new minima found. If this is greater than would be expected from independent data then there is assumed to be a downward trend. A robust line fitting technique is then used to fit a skew line to the points marked as minima. It was shown by Moon et al. [73] that the skew error of Paxson's algorithm is dependent on the absolute skew value - it performs worse for larger skews. Duda et al. [34] also propose a robust least squares regression technique but apply it to all of the data from messages in both directions. They note that the method has the drawback of not explicitly constraining the line to lie fully within the bounds corridor.

Elson et al. [36] describe a protocol called *Reference Broadcast Synchronization* (RBS) which uses linear regression over a window of measurements to estimate skew and offset between nodes of a wireless sensor network, assuming Normally distributed delays. Once a node has determined its skew and offset relative to its neighbours, it broadcasts the information so that the neighbours may then determine their skew and offsets relative to each other. PalChaudhuri et al. [86] provide a closed-form probabilistic bound on the synchronization error for RBS, which is interesting because it allows the derivation of expressions to predict the number of samples required to achieve a certain accuracy. The bounds are based on the assumption that transmit delays are drawn from a Normal distribution, which is a reasonable assumption for wireless sensor networks, but less applicable to Local Area Networks.

A recent development has been the introduction of the IEEE1588 Precision Time Protocol (PTP), which is designed to provide an accurate clock synchronization service over ethernet. The standard requires the use of specialized hardware to generate high quality time stamps. The idea is send two-way timing messages between client

and server at regular intervals, and compute the offset from each message in the same way as Cristian's algorithm [28]. The output from that is then fed into a Proportional + Integral (PI) controller which controls the frequency of the slave clock to drive the offset error to zero.

6.8.3 Maximum Likelihood Estimator

Duda et al. [34] show how to pose the skew estimation problem in a Maximum Likelihood framework. Let $D = \{d_0 \,..\, d_{N-1}\}$ be the set of upper bound offset measurements and $Q = \{q_0 \,..\, q_{M-1}\}$ be the set of lower bound offset measurements. If the data are generated from a pair of clocks with a linear relationship defined by $\alpha t + \beta$ then each point can be thought of as being produced by a probabilistic process whereby $d_i = \alpha t_i + \beta + \delta + w_i$ and $q_j = \alpha t_j + \beta - \delta - w_j$. Here δ is the minimum network delay and $w_i \sim \mathcal{W}$ is a positive random variable representing unknown packet delays. We discuss the form of the distribution $\mathcal{W}$ later.

The proposed likelihood function is

$$L(\hat{\alpha}, \hat{\beta}) = \prod_{i=0}^{N-1} p(w_i = d_i - (\hat{\alpha} t_i + \hat{\beta} + \delta)) \prod_{j=0}^{M-1} p(w_j = (\hat{\alpha} t_j + \hat{\beta} - \delta) - q_j), \qquad (6.22)$$

which is the product of message delay probabilities. The estimator works by choosing $\hat{\alpha}$ and $\hat{\beta}$ to maximize the likelihood function. The estimator will always return a solution that lies entirely between the upper and lower packet delay measurements, because negative delays have zero probability.

The ML estimator is the optimal estimator for the skew estimation problem - it makes best use of the available information. It has two drawbacks which make it inappropriate to use in practice. To solve it requires an expensive iterative batch optimization procedure, making it intractable to run online over long time intervals. It also requires a priori knowledge of the true packet delay distribution and minimum

network delay, which are usually not available.

Noh et al. [82] derive Maximum Likelihood Estimators (MLEs) for offset and skew under the assumption of both Gaussian and Exponential delay distributions. Additionally, they are able to derive the Cramér-Rao Lower Bound for the ML Estimators of offset and skew. The Cramér-Rao Lower Bound gives a bound on the lowest achievable variance of *any* estimator, for a particular likelihood function. The derived ML estimators are computationally expensive, so the authors propose a much simpler approximation which is a function of only the first and last measurement pairs; it ignores all the data in between. It also has the advantage of not requiring a priori knowledge of the delay distribution parameters.

Chaudhari et al.[20] improved on the MLE derivations of Noh et al. by coming up with a joint MLE for offset and skew, in the presence of exponential delays. The resulting algorithm is even more complicated, so again they present an approximation. They find the two measurement pairs with smallest Round Trip Time, and fit a line through them, again ignoring all of the other data. Sometimes the line is ill conditioned, so it may be necessary to replace one of the two measurement pairs with one of the endpoints of the dataset. Later, Chaudhari et al. [19] derived the MLE for a clock offset model allowing constant non-zero drift. The algorithm is, unsurprisingly, more complicated still, but no low-cost approximation is given.

The algorithms we move on to discuss now may also be regarded as ways of approximating the ML Estimate, but arguably they make more effective use of the available data than the approximations just discussed.

6.8.4 Linear Programming Approaches

Linear programming approaches attempt to minimize or maximize an objective function, give a series of constraints imposed by the timing data. The assumption with these is that clocks have zero drift, so that the problem amounts to finding the pa-

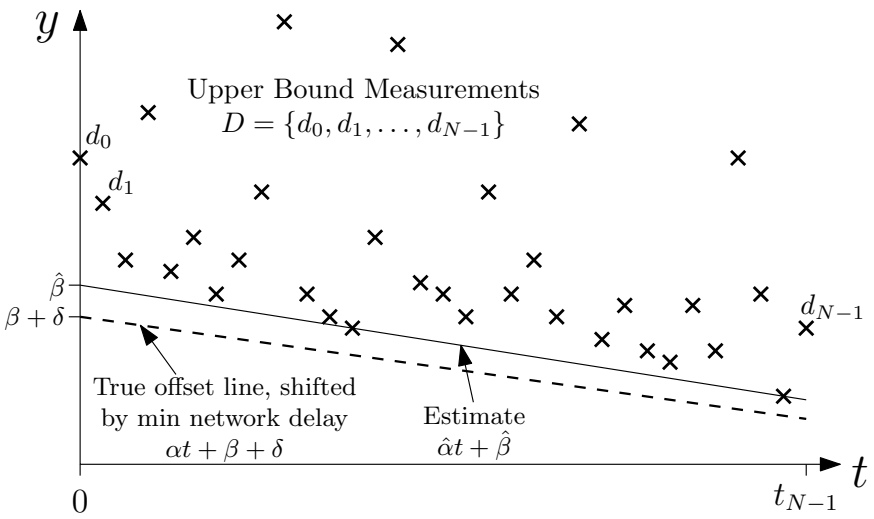


Figure 6.3: The linear programming technique of Moon et al. applies to one-way delay measurements $D = \{d_0 \,..\, d_{N-1}\}$ only, which form an upper bound on the line with true skew α, offset by the minimum network delay, δ. The algorithm finds a line $\hat{\alpha}t + \hat{\beta}$ which touches the underside of the data, but does not lie above any data point.

rameters of a line that best approximates the skew and offset between clocks.

Moon et al. [73, 72] consider only one-way timing information and ensure that the bounds are satisfied by using a linear programming technique to fit a line hard up against the data points. The algorithm is very effective and lends itself to an incremental implementation. Figure 6.3 shows the set of upper bound measurements $D = \{d_0 \,..\, d_{N-1}\}$ generated from a pair of clocks with true skew α and initial offset β. The algorithm chooses its estimate line $\hat{\alpha}t + \hat{\beta}$ to minimize the objective function:

$$\begin{aligned} \min_{\hat{\alpha},\hat{\beta}} \quad & \textstyle\sum_{i=0}^{N-1} \left(d_i - \hat{\alpha} t_i - \hat{\beta} \right) \\ \text{s.t.} \quad & \hat{\alpha} t_i + \hat{\beta} \leq d_i, \qquad i = 0 \,..\, N-1 \end{aligned} \tag{6.23}$$

Moon et al. note that linear programming problems in two dimensions may be solved in $O(N)$ time [68]. They do not offer any information about the convergence properties of their estimator. In Sections 7.1.2 and 7.2 we will examine its convergence properties and compare them with our own clock synchronization algorithm. Because

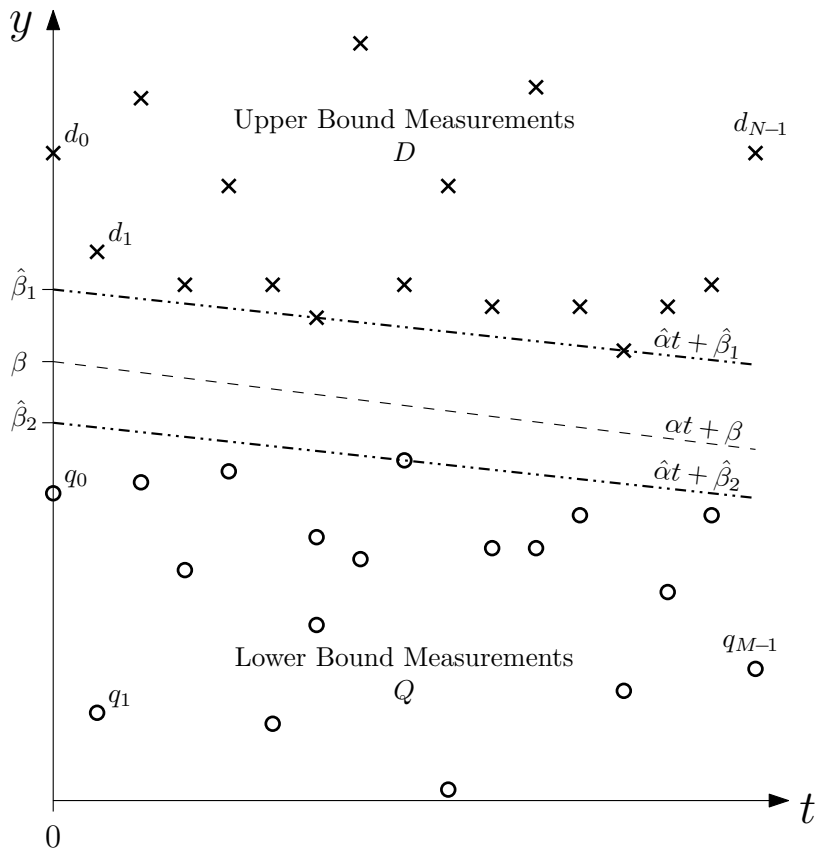


Figure 6.4: The objective function of Sirdey and Maurice finds the two lines of maximum separation that lie within the bounds corridor and share the same slope, $\hat{\alpha}$. The upper and lower lines have intersect $\hat{\beta}_1$ and $\hat{\beta}_2$ respectively.

we will regularly mention the synchronization scheme of Moon et al., we will usually refer to it simply as the *Moon Estimator*.

Much of the next Chapter will be devoted to analyzing an objective function proposed by Sirdey and Maurice [103]. They present a linear programming algorithm which seeks to find the pair of maximally separated lines of equal slope that fit within the data corridor. Figure 6.4 shows an example, with the lines of maximum separation being $\hat{\alpha}t + \hat{\beta}_1$ and $\hat{\alpha}t + \hat{\beta}_2$. If the upper bound measurements are given by $D = \{d_0 \,..\, d_{N-1}\}$ and the lower bound measurements by $Q = \{q_0 \,..\, q_{M-1}\}$ then the

lines of maximum separation are found by maximizing the objective function,

$$\begin{aligned} \max_{\hat{\alpha},\hat{\beta}_1,\hat{\beta}_2} \quad & \hat{\beta}_1 - \hat{\beta}_2 \\ \text{s.t.} \quad & \hat{\alpha}t_i + \hat{\beta}_1 \leq d_i, \qquad i = 0\,..\,N-1 \\ & \hat{\alpha}t'_j + \hat{\beta}_2 \geq q_j, \qquad j = 0\,..\,M-1 \end{aligned} \tag{6.24}$$

Given that the data is expected to form a linear corridor, this approach (over the others discussed) seems to make best use of all the information available and has the effect of minimizing the maximum offset error. Assuming that message delays are symmetric, one would use a line $\hat{\alpha}t + (\hat{\beta}_1 + \hat{\beta}_2)/2$ as the final offset estimate. The reader may recognize similarities between this estimator and the Maximum Margin Classifier of the Support Vector Machine [13].

Sirdey and Maurice show empirically that the method satisfies the stringent synchronization requirements of the GSM Base Station Network, of 50 PPB (parts per billion) in skew, given sufficient time to converge. We will present theoretical convergence results for the algorithm later, and compare it to the Moon Estimator.

The implementation proposed by Sirdey and Maurice involves gathering timing data over a period of around 10 minutes (in which the clocks are expected to exhibit no drift) and then performing a batch linear programming optimization of equation 6.24 in linear time [68, 35]. They suggest that an implementation of the well known simplex algorithm [75] could also be used.

In Section 8.0.1 we will present an algorithm which maximizes the objective function in equation 6.24, but does so much more efficiently than the algorithm proposed by Sirdey and Maurice. We will refer to the resulting estimator as the *MaxSep Estimator*, due to its goal of finding lines of Maximum Separation.

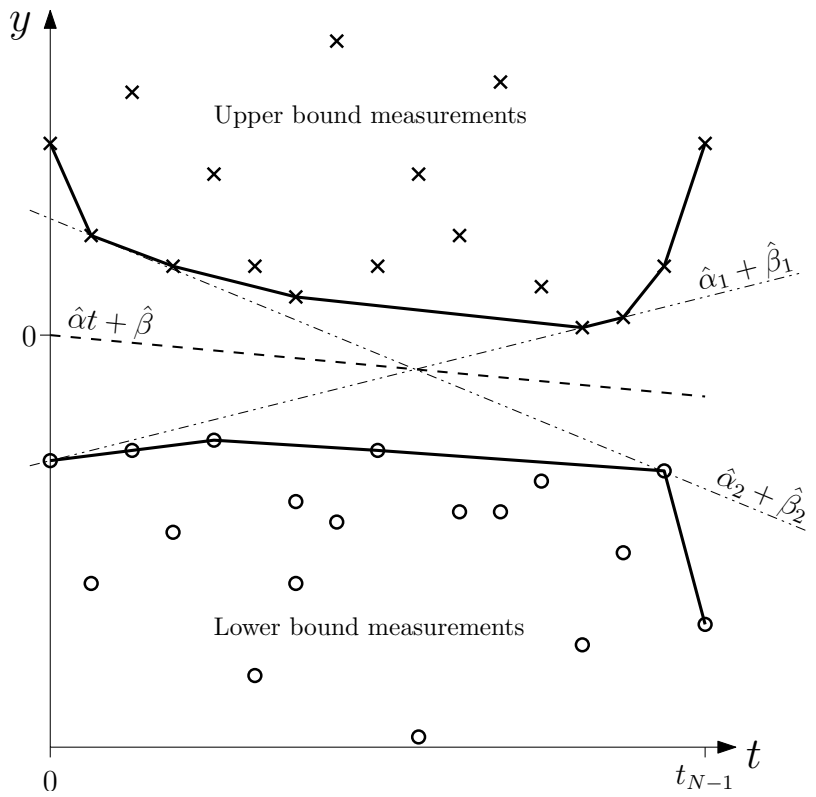


Figure 6.5: The method of Duda et al. After computing upper and lower convex hulls, the pair of lines passing between the hulls with minimum and maximum slope ($\hat{\alpha}_2$ and $\hat{\alpha}_1$ respectively) are found. The estimate $\hat{\alpha}t + \hat{\beta}$ is then given by the line that bisects the two.

6.8.5 Convex Hull approaches

Duda et al. [34] were amongst the earliest to recognize the importance of the convex hull in estimating clock skew. After determining that linear regression gives a poor result, they proposed a convex hull based algorithm. Figure 6.5 demonstrates the idea, which involves computing convex hulls of the upper and lower bound packet traces. By walking the hulls from opposite directions and looking for lines which lie at a tangent to both hulls, a pair of bounding lines passing through the corridor with minimum and maximum slope are found. The estimate is then given by the line that bisects the bounds.

Later, Zhang et al. [123] showed that the Moon objective function (Equation 6.23) could be minimized using a convex hull approach in linear time. It should be no surprise that the convex hull plays a part, since any line which is pushed up against

the data cannot touch any point not on the convex hull. The result of Zhang et al. is discussed in Section 7.1.1, and plays an important part in the later convergence proofs.

6.8.6 Upset Detection

A number of authors relax the assumption of zero clock drift and attempt to track the changing skew, as well as various types of clock upset such as resets. This usually comes down to locating *breakpoints* between which the clocks are assumed to behave in a linear fashion. The majority of piecewise linear techniques are off-line and search for optimal breakpoints considering all of the data.

The types of upset considered are

Resets Where a clock has its time reset, but the frequency remains constant.

Hiccups Where a reset is closely followed by an equal and opposite reset, bringing the clock back to its original offset.

Frequency Resets Where the clock frequency is reset, but the offset does not jump.

Drift The frequency varies smoothly over a period of time. This is typical behaviour of a clock under the control of NTP.

Zhang et al. [123] propose offline approaches for detecting clock resets in one-way timing data. They begin with an algorithm which assumes only a single reset point and constant skew. They show how to estimate skew if the reset point is known. In practice the reset point is located by running the algorithm for all possible reset points, and finding the one the best explains the data. Then they generalize the approach to allow an arbitrary number of resets, with a constraint on how regularly they may occur. This involves trying *all possible* combinations of reset points; an $O((N/R)^R NR)$ algorithm for R resets and N samples. In order to reduce the search

time, they employ a divide-and-conquer heuristic, splitting the data into fixed-length intervals and merging adjacent intervals which appear to share the same slope. The single breakpoint algorithm is then applied to each remaining interval. The constraint on the minimum time between resets means that hiccups may not be adequately detected.

A similar approach is used to find piecewise linear skew lines, when there are a finite number of skew adjustment points. The authors do not present an algorithm capable of detecting both resets and skew adjustments simultaneously.

Bi et al. [12] take another offline divide-and-conquer approach, which operates recursively on one-way timing data. First the Moon Estimator is applied to all of the data. If the line does not fit the data well then the data is divided in two and the algorithm called recursively on each segment. the measure of line fit goodness is to count the number of delay measurements that touch the fitted line. A probabilistic threshold must be chosen to tune this. the authors note that the algorithm can fail when the minimum delay measurements are not fairly distributed. They provide a solution to the problem by applying the line goodness test at multiple scales on each segment, resulting in an $O(N \log_2 N)$ algorithm.

The advantages of the technique are that it can detect an arbitrary number of clock upsets without increasing complexity. The disadvantages are that it has a number of thresholds that must be chosen for window lengths and minimum delay measurement density. It will also tend to overfit to network load fluctuations, unless the minimum segment length is kept relatively large.

Wang et al. [115] provide another divide-and-conquer approach, again on one-way measurements. They find stable clock periods (constant skew) and fit piecewise linear segments, using a recursive top-down algorithm. The novelty of their method is in a second pass which can undo previous poor segmentation decisions by re-merging segments. The algorithm is $O(KN^2)$, where K is the final number of segments.

To enable online use, the authors propose running the segmentation over a sliding window. this will inevitably lead to a delay equal to the window period, and would result in an $O(Kw^2N^2)$ algorithm, where w is the window length.

A problem with many divide and conquer techniques is that since the algorithm is offline and must be presented with all of the data, sometimes clock upsets are detected *before* they occur. In one example presented by Bi et al., a clock hiccup is detected 10-15 seconds in advance, and the return to normal time is not detected for another 10-15 seconds.

Khlifi and Gregoire [55] suggest using a sliding window and exponential smoothing on the skew estimates (which will add further phase lag) as well as a number of heuristics to condition the data.

6.9 Chapter Summary

We have discussed the fundamentals of the clock synchronization problem and the methods used to model the offset and skew relationships between different clocks. We have shown how packet delay measurements between two networked computers can be represented as a 'corridor' which gives upper and lower bounds on the offset between the clocks of the computers.

We explored a number of existing methods of analyzing the corridor data in order to recover the skew and offset between clocks, focusing particularly on linear programming and convex hull based methods, which make most efficient use of historical timing data. We then investigated methods of detecting clock 'upsets', finding that existing approaches tend to require expensive offline processing, or put constraints on the type and regularity of upsets.

In the following chapters we look more in depth at the linear programming approaches and will, from the insights gained, develop a highly efficient clock synchro-

nization algorithm capable of detecting and adjusting to upsets online, in near-optimal time. Our motivation remains the generation of accurate timestamps across a distributed system.

Chapter 7

A Probabilistic Analysis

In this Chapter, continuing our pursuit of a distributed time-synchronisation system, the performance of two estimators is studied in depth; the Moon Estimator and the MaxSep Estimator. The originating authors of the two techniques present only empirical results for the convergence properties of their estimators. We provide convergence proofs and develop probabilistic convergence models for both. Those results give us a basis for comparing the two estimators and for deriving real-time performance measures.

7.1 Moon Estimator

Moon et al. [73, 72] do not present any form of convergence proof for their Linear Programming based approach to clock synchronization (henceforth referred to as the Moon Estimator). In this section, we present a new proof of convergence, and develop a probabilistic model to characterize the convergence properties. The model will form a basis for comparing the performance of different objective functions.

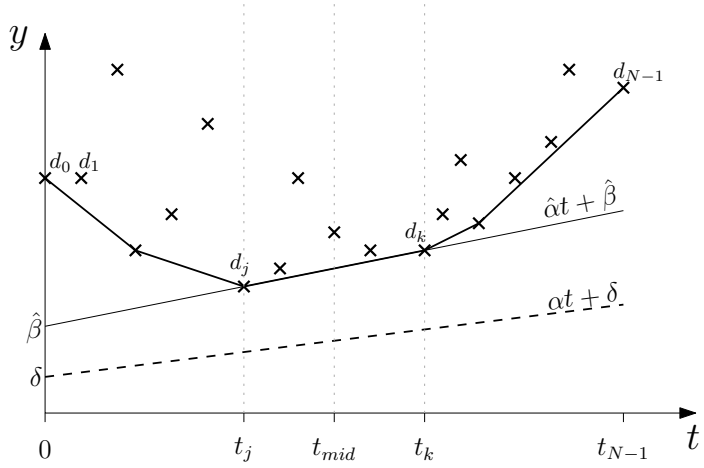


Figure 7.1: A set of upper bound delay measurements shown with their convex hull. The Moon objective function is minimized by the hull segment spanning the mean measurement time, $t_{mid} = \frac{1}{N}\sum_{i=0}^{N-1} t_i$. In this case that segment is $\{d_j, d_k\}$.

7.1.1 A Dual Formulation for the Moon Estimator

Figure 7.1 shows a set of upper bound offset measurements, $D = \{d_0 \,..\, d_{N-1}\}$ taken between a pair of clocks. The true slope of the measurements is α. Note that we have chosen the true intersect value to be $\beta = 0$ in order to simplify the following steps, though no loss of generality is incurred. With a fixed sample period T and a minimum message propagation delay, δ, we model the measurements with

$$d_i = \alpha T i + \delta + w_i, \qquad i = 0 \,..\, N-1 \tag{7.1}$$

where $w_i \sim \mathcal{W}$ is a random variable representing unknown packet delays. Packet delays are *never* negative, so w_i is guaranteed to be positive. All measurements, $d_i \in D$ must lie above the line $\alpha t + \delta$.

Recall that the Moon [73] objective function is

$$\begin{aligned} \min_{\hat{\alpha},\hat{\beta}} \quad & \textstyle\sum_{i=0}^{N-1} \left(d_i - \hat{\alpha} t_i - \hat{\beta} \right) \\ \text{s.t.} \quad & \hat{\alpha} t_i + \hat{\beta} \leq d_i, \qquad i = 0 \,..\, N-1 \end{aligned} \tag{7.2}$$

Zhang et al. [123] showed that minimizing the objective function may be achieved using a convex hull approach. A convex hull is shown fitted beneath the points in Figure 7.1. Notice that no line lying below the data points can intersect the convex hull, and that if such a line passes through more than one data point then it must be collinear with one of the hull segments. We will use the short notation $\{d_j, d_k\}$ to refer to a section of the hull joining points (t_j, d_j) and (t_k, d_k).

Zhang et al. demonstrated that minimizing the Moon Objective Function is completely equivalent to choosing the hull segment spanning the time $t_{mid} = \frac{1}{N}\sum_{i=0}^{N-1} t_i$ to be the line estimate. In Figure 7.1, the segment $\{d_j, d_k\}$ is the one that spans t_{mid}, and is thus the estimate $\hat{\alpha}t + \hat{\beta}$. As N increases, the number of low-latency packets seen will increase, and the estimate will approach $\alpha t + \beta + \delta$.

The Moon Estimator lends itself to an incremental implementation because the convex hull can be computed incrementally, and maintaining a pointer to the hull segment spanning t_{mid} is trivial. A suitable algorithm for computing the hull is reproduced in Section 8.0.1.

Before we move on, consider the case when t_{mid} coincides with a point on the hull; there is no single line which minimizes the objective function. Zhang et al. do not discuss this case, but the set of possible solutions will be all lines passing through d_{mid}, with slope bounded by the slopes of the two adjacent hull segments. In practice a sensible course of action would be to take the line passing along the later hull segment, since the constantly marching mid-point will shortly lie in that segment.

7.1.2 Proof Of Convergence For The Moon Objective Function

Having a convex hull based formulation for the Moon Estimator allows us to apply the tools of convex analysis and computational geometry and gain useful insights into the performance of the Estimator. Moon et al. show only empirically that the estimate produced by their method converges to the true answer. We now present a

proof of convergence.

Suppose that after N measurements, the convex hull segment spanning t_{mid} lies on the interval $[t_j, t_k]$ where $t_j < t_{mid} < t_k$.

Lemma 7.1.1 *Any hull segment $\{d_r, d_s\}$ lies on or above the line $\alpha t + \delta$ within the interval $[t_r, t_s]$.*

Proof The measurements d_r and d_s form the endpoints of the hull segment. Since $d_i \geq \alpha t + \delta$ for all i, the hull segment must lie on or above the line $\alpha t + \delta$. ∎

Lemma 7.1.2 *If the hull segment $\{d_j, d_k\}$ has slope $\hat{\alpha} \geq \alpha$ then*

$$\hat{\alpha} T i + \hat{\beta} \geq \alpha T i + \delta, \qquad i = k \,..\, N-1 \tag{7.3}$$

Proof Follows from Lemma 7.1.1. If the line $\hat{\alpha} t + \hat{\beta}$ is above or touching $\alpha t + \delta$ at time t_k then it must remain above or touching at all subsequent times if $\hat{\alpha} \geq \alpha$. ∎

Lemma 7.1.3 *If some hull segment $\{d_r, d_s\}$ has slope $a < \alpha$ then as $N \to \infty$ no future segment spanning t_r will have slope greater than a.*

Proof It is a property of the convex hull that every segment has a slope strictly greater than that of all previous segments. When a new data point p is added on the right, the hull changes in one of two ways. If p lies above the last hull segment line then a new, steeper segment is appended to the hull, with p as its end point. If the new point lies on or below any previous hull segment lines then those segments are removed from the hull and then a new segment (less steep than all those removed) is appended with p as its end point. Therefore existing segments are only modified by a merge operation, and the merge operation only ever replaces old hull segments with a less steep segment than before.

Thus if the segment $\{d_r, d_s\}$ is replaced in a merge operation, the replacement must both span t_r and have new slope $a' \leq a$. ∎

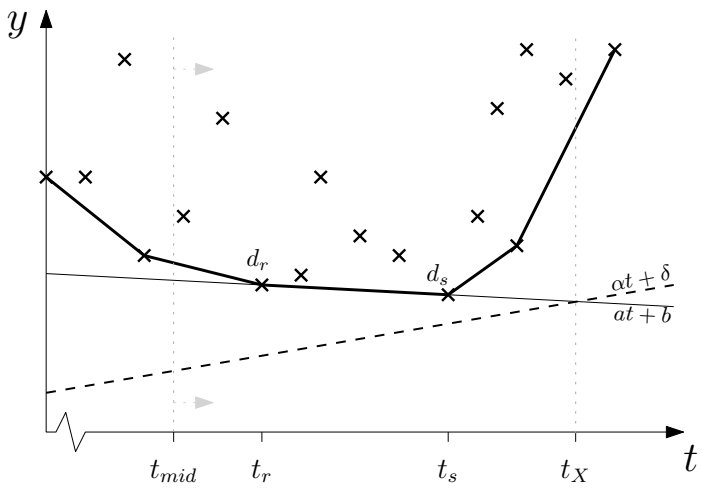


Figure 7.2: If a hull segment has slope less than α then the line through that segment will intersect the true line at some future time, t_X. As the number of samples increases and t_{mid} progresses to the right, it will eventually pass t_X. After that time, no hull segment spanning t_r will also span t_{mid}.

Lemma 7.1.4 *If some hull segment $\{d_r, d_s\}$ has slope $a < \alpha$ then as $N \to \infty$, the hull segment spanning time t_{mid} will not span time t_r*

Proof Figure 7.2 shows a situation where a hull segment $\{d_r, d_s\}$ with slope $a < \alpha$ lies after t_{mid}. As N increases, t_{mid} advances to the right at half the rate. Since $a < \alpha$, the intersection point t_X between the extended hull segment and the line $\alpha t + \delta$ must lie after t_r. From Lemma 7.1.3, any future hull segment spanning time t_r cannot have slope greater than a, thus its intersection point with the line $\alpha t + \delta$ must be *on* or *to the left of* t_X. Since no hull segment may intersect the line $\alpha t + \delta$ (Lemma 7.1.1), no segment spanning t_r will extend beyond t_X. As $N \to \infty$, $t_{mid} > t_X$, so a segment spanning t_{mid} cannot also span t_r. ∎

Lemma 7.1.5 *In the limit as the number of samples tends towards infinity, the length of the convex hull segment spanning t_{mid} is infinite.*

Proof Consider the case shown in Figure 7.3 when the hull segment $\{d_j, d_k\}$ spanning t_{mid} is followed by a segment $\{d_k, d_l\}$ with equation $\hat{\alpha}'t + \hat{\beta}'$.

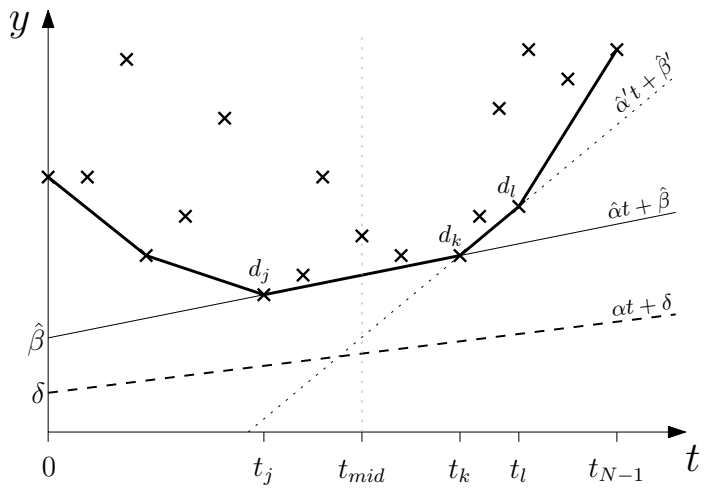


Figure 7.3: As the number of samples tends to infinity, the length of the segment spanning t_{mid} (in this case $\{d_j, d_k\}$) also tends to infinity.

The existence of the hull segment $\{d_k, d_l\}$ implies that all data points d_i for $k < i < N$ lie on or above the line through that segment, $\hat{\alpha}'t + \hat{\beta}'$, and therefore strictly above $\hat{\alpha}t + \hat{\beta}$. The probability of drawing a set of measurements whose random delays satisfy that constraint is given by

$$P(\hat{\alpha}' > \hat{\alpha}) = \prod_{i=k+1}^{N-1} P(d_i > \hat{\alpha}Ti + \hat{\beta}) \tag{7.4}$$

$$= \prod_{i=k+1}^{N-1} P(\alpha Ti + \delta + w_i > \hat{\alpha}Ti + \hat{\beta}) \tag{7.5}$$

$$= \prod_{i=k+1}^{N-1} P(w_i > (\hat{\alpha}Ti + \hat{\beta}) - (\alpha Ti + \delta)) \tag{7.6}$$

where w_i represents a random positive delay, so $w_i \geq 0$ for all i. Finding the limit as the number of samples tends to infinity gives

$$\lim_{N\to\infty} P(\hat{\alpha}' > \hat{\alpha}) = \begin{cases} 1 & (\hat{\alpha}Ti + \hat{\beta}) < (\alpha Ti + \delta), \qquad \forall \quad i > k \\ 0 & \text{otherwise} \end{cases} \tag{7.7}$$

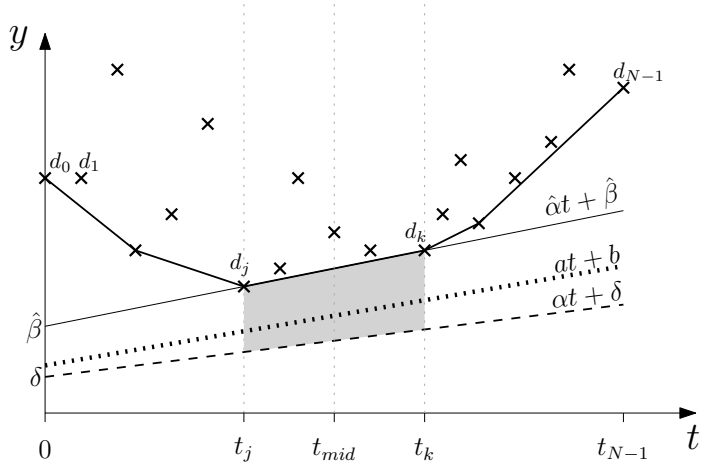


Figure 7.4: Any line passing through both ends of the shaded area must tend towards $\alpha t + \delta$ as the number of samples tends to infinity.

The first case would be satisfied only when $\hat{\alpha} < \alpha$ (from Lemma 7.1.1) and implies intersection with the line $\alpha t + \delta$ at time t_k. From Lemma 7.1.4, hull segment $\{d_j, d_k\}$ would therefore not span t_{mid} in the limit as $N \to \infty$. Thus the first case is never satisfied.

Therefore, in the limit as $N \to \infty$, there can be no segment after $\{d_j, d_k\}$ that has greater slope. The convex hull does not allow for subsequent segments with lesser or equal slope, so it must be the case that $k \to \infty$ as $N \to \infty$. ∎

Lemma 7.1.5 offers good justification of our forthcoming assertion (Section 8.0.1) that in practice convex hulls rarely grow above 20 segments when applied to packet delay data. The probability of the hull growing in size without bound is vanishingly small.

Lemma 7.1.6 *If there is a line, $at + b$ satisfying $\alpha t + \delta \leq at + b \leq \hat{\alpha}t + \hat{\beta}$, for $t_j \leq t \leq t_k$ spanning t_{mid} then it must converge to the line $\alpha t + \delta$ as $N \to \infty$*

Proof Figure 7.4 shows a shaded area corresponding to the feasible region for the line $at + b$. The probability that all data in the interval $[t_j, t_k]$ lies above the line is

given by

$$F(a,b) = \prod_{i=j}^{k} P(d_i \geq aTi + b) \tag{7.8}$$

$$= \prod_{i=j}^{k} P(w_i \geq (aTi + b) - (\alpha Ti + \delta)) \tag{7.9}$$

In the limit as $N \to \infty$, Lemma 7.1.5 tells us that $k \to \infty$, so

$$\lim_{N\to\infty} F(a,b) = \lim_{k\to\infty} F(a,b) \tag{7.10}$$

$$= \prod_{i=j}^{\infty} P(w_i \geq (aTi + b) - (\alpha Ti + \delta)) \tag{7.11}$$

$$= \begin{cases} 0 & aTi + b > \alpha Ti + \delta, \qquad j \leq i \leq \infty \\ 1 & aTi + b \leq \alpha Ti + \delta, \qquad j \leq i \leq \infty \end{cases} \tag{7.12}$$

If $at + b$ passes through the shaded area it cannot lie below $\alpha t + \beta$ in the interval $[t_j, t_k]$, so the only possible outcome is that $aTi + b = \alpha Ti + \delta$ for $j \leq i \leq \infty$. ∎

Using the Lemmas proved in this section, we are now in a position to present the key convergence proof for the Moon Estimator:

Theorem 7.1.7 *The Moon Estimator converges to the line $\alpha t + \delta$ as $N \to \infty$.*

Proof The Moon Estimator picks as its estimate a line passing along the hull segment which spans t_{mid}. This line is guaranteed to run along the top of the feasible area defined in Lemma 7.1.6 (shaded area in Figure 7.4). Therefore as $N \to \infty$ the estimate must converge to $\alpha t + \delta$. ∎

7.2 Moon Estimator Convergence Properties

In order to model the convergence properties of the Moon estimator, we now derive a hypothetical estimator which we will show to have slope greater than or equal to

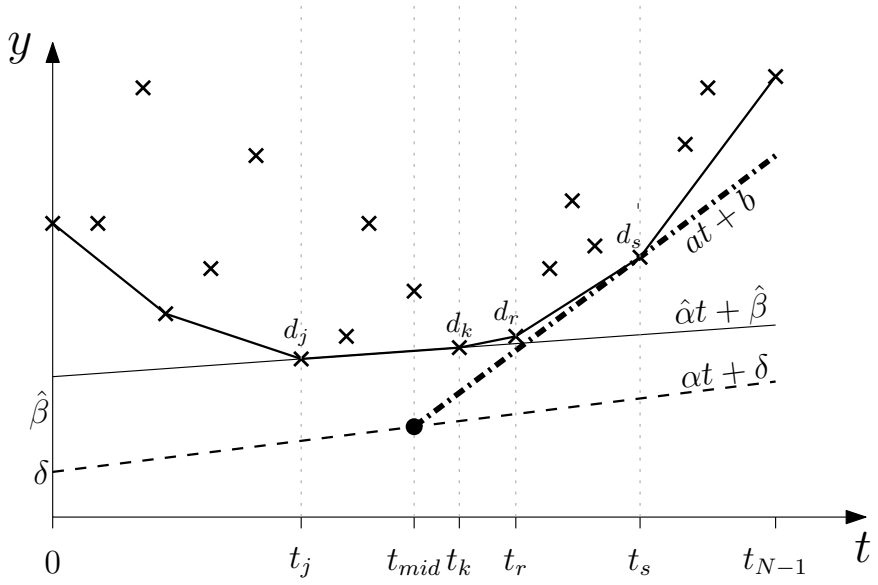


Figure 7.5: The anchored estimator is a line crossing $\alpha t + \delta$ at t_{mid} and lying at a tangent to the right half of the hull. Represented here as a thick dash-dotted line with equation $at + b$, it is guaranteed to be at least as steep as the Moon Estimator at all times.

that of the Moon Estimator at all times. We'll call it the *Anchored Estimator*. It is hypothetical because to implement it would require a priori knowledge of the true clock skew.

Consider Figure 7.5 which shows upper bound packet delay measurements with their convex hull. The Moon Estimate is the line $\hat{\alpha}t + \hat{\beta}$, which passes along the hull segment spanning t_{mid}. Also shown is a half-line $at + b$, which passes through the line of true slope $(\alpha t + \delta)$ at t_{mid}. Its slope is such that it lies at a tangent to the hull right of t_{mid}.

Theorem 7.2.1 *The line $at + b$ must be at least as steep as the Moon estimate, $\hat{\alpha}t + \hat{\beta}$.*

Proof If $at + b$ touches the hull at t_s, at the right hand end of the segment $\{d_r, d_s\}$, then d_r must be above $at + b$, and $at + b$ must be at least as steep as the segment $\{d_r, d_s\}$. $\{d_r, d_s\}$ is right of $\{d_j, d_k\}$ and is steeper, so $at + b$ is at least as steep as $\hat{\alpha}t + \hat{\beta}$. ∎

This is a key result. It tells us that *no matter* what slope the Moon Estimate takes, the Anchored Estimator will *always* be at least as steep. If we can obtain an expression for the convergence properties of the Anchored Estimator then we know that the Moon Estimator must converge at least as rapidly as that.

7.2.1 Anchored Estimator Convergence Properties

For the anchored estimator to have a slope error ε_{anc} strictly greater than some value ϕ, then all measurements must lie above the line $at+b$, with $a = \alpha+\varepsilon_{anc}$. We express this as:

$$P(\varepsilon_{anc} > \phi) = \prod_{i=m}^{N-1} P(d_i > (\alpha+\phi)Ti + b), \qquad m = \left\lceil \frac{N-1}{2} \right\rceil \tag{7.13}$$

Since $d_i = \alpha Ti + \delta + w_i$ (where w_i is a positive random variable representing packet delay) then

$$P(\varepsilon_{anc} > \phi) = \prod_{i=m}^{N-1} P(\alpha Ti + \delta + w_i > (\alpha+\phi)Ti + b) \tag{7.14}$$

$$= \prod_{i=m}^{N-1} P(w_i > \phi Ti + b - \delta) \tag{7.15}$$

but because $(\alpha+\phi)t + b$ intersects $\alpha t + \delta$ at t_{mid},

$$b = -\phi t_{mid} + \delta \tag{7.16}$$

so

$$P(\varepsilon_{anc} > \phi) = \prod_{i=m}^{N-1} P(w_i > \phi(Ti - t_{mid})) \tag{7.17}$$

$$= \prod_{i=0}^{m-1} P(w_i > \phi Ti), \tag{7.18}$$

since $t_{mid} = \frac{N-1}{2}T$ and $m = \left\lceil \frac{N-1}{2} \right\rceil$.

In the standard Cumulative Density Function (CDF) form, this is

$$P(\varepsilon_{anc} \leq \phi) = 1 - \prod_{i=0}^{m-1} \left[1 - P(w_i \leq \phi T i)\right], \tag{7.19}$$

since $t_{mid} = \frac{N-1}{2}T$ and $m = \left\lceil \frac{N-1}{2} \right\rceil$.

7.2.2 Moon Estimator Convergence Bound

The Anchored Estimate, $at + b$ is *guaranteed* (by Lemma 7.2.1) to be at least as steep as the Moon Estimate, $\hat{\alpha}t + \hat{\beta}$, so it must be the case that

$$P(\varepsilon_{Moon} \leq \phi) \geq P(\varepsilon_{anc} \leq \phi), \tag{7.20}$$

leading us to the main result of this section, which is a bound on the probability distribution of slope error for the Moon Estimator:

$$\boxed{P(\varepsilon_{Moon} \leq \phi) \geq 1 - \prod_{i=0}^{m-1} \left[1 - P(w_i \leq \phi T i)\right]} \tag{7.21}$$

7.3 MaxSep Estimator

In this section we investigate the objective function proposed by Sirdey and Maurice [103], deriving similar convergence bounds as we have just done for the Moon Estimator.

The objective function of Sirdey and Maurice amounts to finding two maximally separated lines of equal slope, which lie between the upper and lower packet delay

measurements. It is stated as

$$\begin{aligned} \max_{\hat{\alpha},\hat{\beta}_1,\hat{\beta}_2} \quad & \hat{\beta}_1 - \hat{\beta}_2 \\ \text{s.t.} \quad & \hat{\alpha}t_i + \hat{\beta}_1 \leq d_i, \qquad i = 0\,..\,N-1 \\ & \hat{\alpha}t'_j + \hat{\beta}_2 \geq q_j, \qquad j = 0\,..\,M-1 \end{aligned} \tag{7.22}$$

We will refer to the algorithm solving the objective function as the *MaxSep Estimator*.

7.3.1 Some necessary properties

We will shortly show that the MaxSep Objective Function can be maximized using a Convex Hull based formulation of the problem. The steps we follow are similar to those used by Zhang et al. [123], where they show how the Moon objective function can be solved using convex hulls. However, the proofs we present are new. We commence by proving some simple results which will be used to prove the main theorem.

Lemma 7.3.1 *The sum of a pair of convex functions is also a convex function.*

Proof A convex function is characterized by monotonically non-decreasing slope. If $f(t), g(t)$ are convex functions and $h(t) = f(t) + g(t)$ then the slope is given by $h'(t) = f'(t) + g'(t)$, which must also be monotonically non-decreasing. ∎

A corollary of this is that the sum of two concave functions is also concave.

Lemma 7.3.2 *Given a set of coefficients* $\{a_1\,..\,a_n\} \in \mathbb{R}$ *and* $\{b_1\,..\,b_n\} \in \mathbb{R}$*, the function* $f(t) = \min_i(a_i t + b_i)$ *is concave.*

Proof See Figure 7.6. Suppose that on the interval $[t_p, t_q]$ the line minimizing $f(t)$ is $a_j t + b_j$ and on the interval $[t_q, t_r]$ the minimizing line is $a_k t + b_k$, with $t_p < t_q < t_r$.

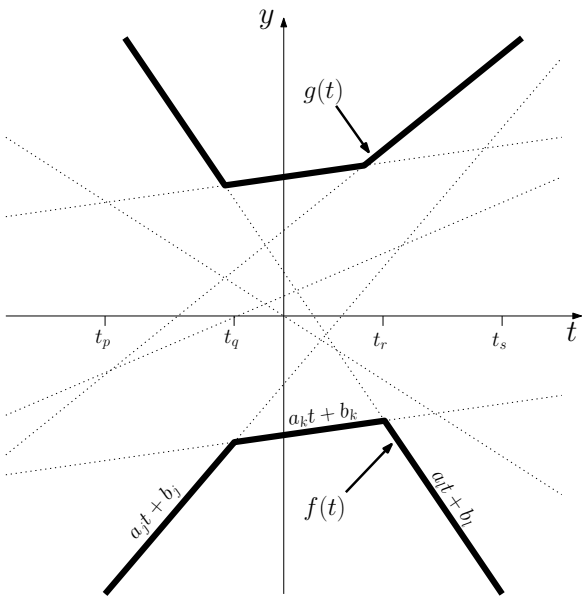


Figure 7.6: For a set of coefficients $\{a_1 \,..\, a_n\} \in \mathbb{R}$ and $\{b_1 \,..\, b_n\} \in \mathbb{R}$, the function $f(t) = \min_i(a_i t + b_i)$ is concave and $g(t) = \max_i(a_i t + b_i)$ is convex.

We may state that,

$$t_p \leq t \leq t_q : \qquad a_j t + b_j \leq a_k t + b_k$$

$$(a_j - a_k)t \leq b_k - b_j \tag{7.23}$$

$$t_q \leq t \leq t_r : \qquad a_j t + b_j \geq a_k t + b_k$$

$$(a_j - a_k)t \geq b_k - b_j \tag{7.24}$$

so that

$$(a_j - a_k)t_p \leq (a_j - a_k)t_r \tag{7.25}$$

Since $t_r > t_p$, this inequality is only satisfied when $a_j > a_k$. The slope of $f(t)$ is therefore monotonically non-increasing and $f(t)$ is concave. ∎

A corollary of this is that the function $g(t) = \max_i(a_i t + b_i)$ is convex. Figure 7.6 gives a graphical representation of both functions.

Lemma 7.3.3 *The minimum vertical distance between upper and lower convex functions is found when both functions take the same gradient at the same instant.*

Proof If $f(t), g(t)$ are upper and lower convex functions respectively, then the distance between them is given by $d(t) = f(t) - g(t)$. The time, t_{min} at which the minimum distance occurs is found by taking the derivative and setting it to zero, so that $f'(t_{min}) = g'(t_{min})$. ∎

If a line lies entirely beneath a continuous, differentiable convex function except for a single point of contact, then it must be tangential to the convex function at that point. It will therefore have the same gradient. For the case of a convex hull, which is a piecewise linear function, the contact point may be a vertex, at which the gradient is discontinuous. In this case, the line must have a slope bounded by the gradients of the two adjacent hull segments (or one of $\{-\infty, \infty\}$ if the vertex is an endpoint). For convenience, we will treat the vertices of the hull as being able to take any gradient between the bounds of the two adjacent segments.

7.3.2 The dual formulation

We are now in a position to show how the MaxSep objective function may be solved efficiently using a convex hull approach. As before, let $D = \{d_0, d_1, \ldots, d_{N-1}\}$ be the set of upper bound measurements and $Q = \{q_0, q_1, \ldots, q_{M-1}\}$ be the set of lower bound measurements. Assume that we have available to us a pair of lower and upper convex hulls fitted to the two sets of measurements.

The lines of maximum separation found by the objective function in Equation 6.24 must both lie between the two convex hulls. The upper line, $\hat{\alpha}t + \hat{\beta}_1$ must contact the upper hull, and the lower line, $\hat{\alpha}t + \hat{\beta}_2$ must contact the lower hull. The following steps will prove that the point of contact for both lines must occur where the hulls have minimum vertical separation.

We define a function

$$
\begin{aligned}
f(\hat{\alpha}) = \max_{\hat{\beta}_1, \hat{\beta}_2} \quad & \hat{\beta}_1 - \hat{\beta}_2 \\
\text{s.t.} \quad & \hat{\alpha} t_i + \hat{\beta}_1 \leq d_i, \qquad i = 0 \,..\, N-1 \\
& \hat{\alpha} t'_j + \hat{\beta}_2 \geq q_j, \qquad j = 0 \,..\, M-1
\end{aligned}
\tag{7.26}
$$

which is the value of the MaxSep objective function when the slope $\hat{\alpha}$ is provided. That is, it finds the two lines of maximum separation which have the specified slope, $\hat{\alpha}$.

Lemma 7.3.4 *$f(\hat{\alpha})$ is concave in $\hat{\alpha}$*

Proof Given $\hat{\alpha}$, the function is maximized by pushing the lines $\hat{\alpha}t + \hat{\beta}_1$ and $\hat{\alpha}t + \hat{\beta}_2$ apart until they contact data above and below, so that $\hat{\beta}_1$ and $\hat{\beta}_2$ are given by

$$
\begin{aligned}
\hat{\beta}_1 &= \min_i (d_i - \hat{\alpha} t_i) \\
\hat{\beta}_2 &= \max_j (q_j - \hat{\alpha} t'_j) \\
&= -\min_j (\hat{\alpha} t'_j - q_j)
\end{aligned}
$$

and the function becomes

$$
f(\hat{\alpha}) = \min_i (-\hat{\alpha} t_i + d_i) + \min_j (\hat{\alpha} t'_j - q_j). \tag{7.27}
$$

By Lemma 7.3.2 the min function is concave, so by Lemma 7.3.1, $f(\hat{\alpha})$ must also be concave. ∎

Figures 7.7 and 7.8 illustrate the concavity of the function $f(\hat{\alpha})$ for an example dataset.

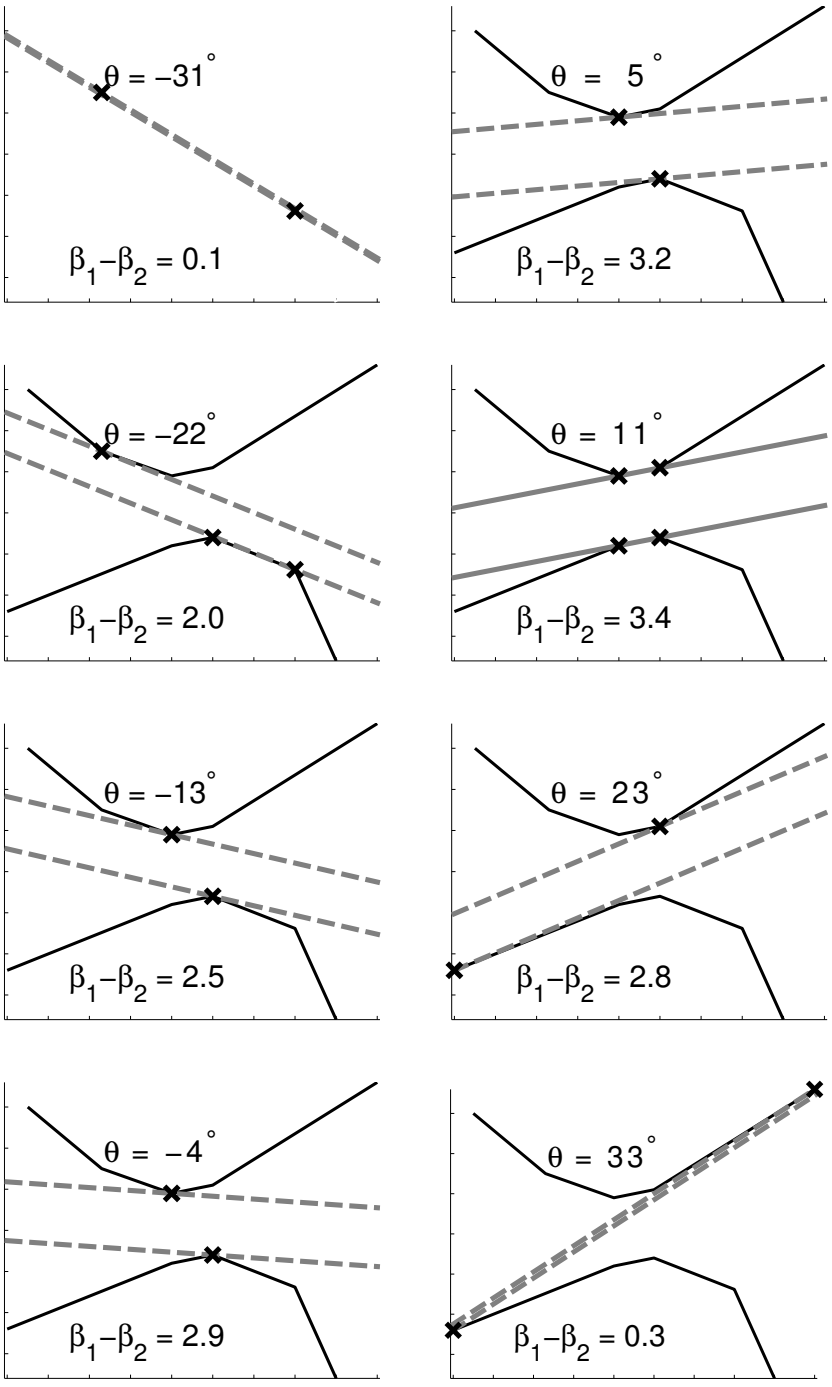


Figure 7.7: Maximally separated lines for a range of slope values, where $\theta = \arctan(\alpha)$. Contact points are shown as crosses. Two crosses on a hull show that a whole segment is in contact. Notice that as the slope increases, the point of contact on the upper hull moves *right* and the point of contact on the lower hull moves *left*. The position at which the contact points coincide is the point at which the lines have maximum separation. It also corresponds to the minimum vertical distance between the two hulls. In this example, it occurs when the lines have angle $\theta = 11°$.

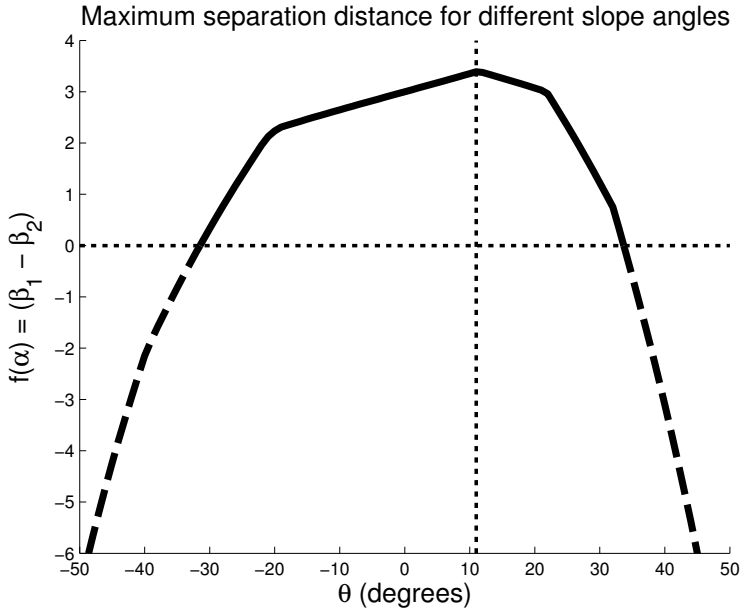


Figure 7.8: The objective function of equation 7.26 is concave and piecewise linear. It represents the distance between the maximally separated lines, for a given slope value $\hat{\alpha}$. This example is plotted from the case shown in Figure 7.7. When the function takes values below zero it violates the constraint that the lines of maximum separation must lie between the convex hulls, because it implies that $\hat{\beta}_1 < \hat{\beta}_2$.

Lemma 7.3.5 *The MaxSep Objective Function is maximized by choosing the pair of equal slope lines which contact the top and bottom convex hulls at the same moment in time.*

Proof Consider two inputs to the function f, a_1 and a_2, with $a_1 < a_2$. If both inputs cause the lines of maximum separation to pass through the same hull points then we have the situation illustrated in Figure 7.9. Evaluating the function at both inputs yields

$$\begin{aligned} f(a_1) &= -(a_1 t_* - d_*) - (q_* - a_1 t'_*) & f(a_2) &= -(a_2 t_* - d_*) - (q_* - a_2 t'_*) \\ &= a_1(t'_* - t_*) + d_* - q_* & &= a_2(t'_* - t_*) + d_* - q_* \end{aligned} \tag{7.28}$$

Because f is concave then any local maximum solution must also be the global max-

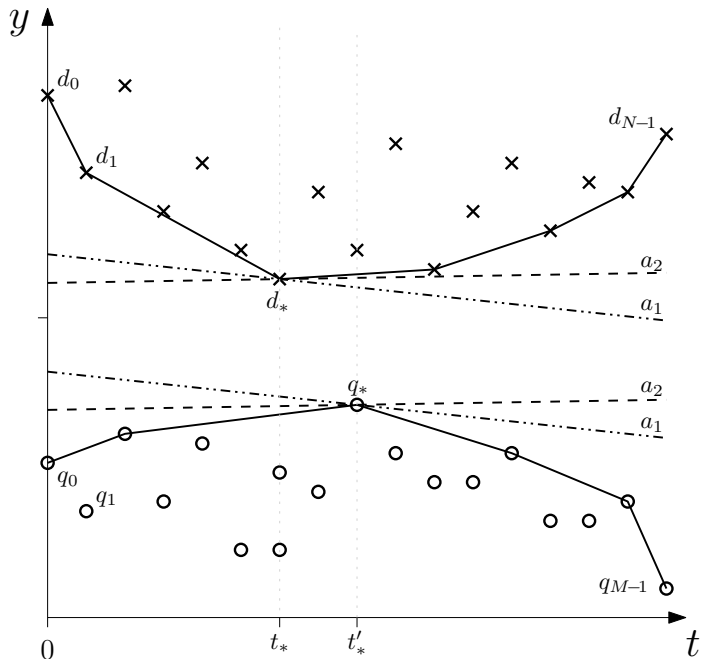


Figure 7.9: Finding the hull vertices through which the maximally separated lines pass. Here the function f is evaluated at two values, a_1 and a_2, with $a_1 < a_2$ such that the lines of maximum separation pass through the same hull points, (t_*, d_*) and (t'_*, q_*).

imum solution, for which we now search. From equation 7.28,

$$f(a_1) - f(a_2) = (a_1 - a_2)(t'_* - t_*) \tag{7.29}$$

$$f(a_1) < f(a_2) \iff t'_* > t_* \tag{7.30}$$

since $a_1 < a_2$. Therefore we may write that

$$\begin{aligned} t'_* > t_* &\iff f'(\hat{\alpha}) > 0 \\ t'_* < t_* &\iff f'(\hat{\alpha}) < 0. \end{aligned} \tag{7.31}$$

The objective function is maximized when $t'_* = t_*$. Thus the objective function is maximized by lines which have slope such that they contact the two hulls at the same moment in time. ∎

An intuition into this result may be gained by studying Figure 7.7.

Lemma 7.3.6 *The lines of maximum separation will contact the hulls where both the upper and lower hull share the same slope.*

Proof The lines of maximum separation are defined as having identical slope. They must both contact their respective hulls at points where the hull has gradient equal to the line slope. Therefore the contact points on both hulls must share the same gradient. ∎

Theorem 7.3.7 *The MaxSep Objective Function is maximized by choosing a pair of lines that contact the convex hulls at the time corresponding to the minimum vertical distance between the hulls.*

Proof By Lemmas 7.3.6 and 7.3.5, the lines of maximum separation correspond to a pair of equal slope lines contacting their respective hulls at the point where the hulls have equal slope at the same time instant. By Lemma 7.3.3, this is the point of minimum vertical distance between the two hulls. ∎

This is an important result, which will (in Section 8.0.1) directly allow us to create an efficient algorithm for solving the MaxSep Objective Function. Intuitively the result makes sense. The lines of maximum separation must pass entirely between the two hulls, including the parts of the hull having minimum vertical distance. Therefore the lines of maximum separation cannot be further apart than the smallest vertical distance between the two hulls.

7.4 MaxSep Estimator Convergence

Using the results from the previous section, we are now in a position to derive convergence properties for the MaxSep Estimator, as we did for the Moon Estimator. First we present a proof of convergence, which follows directly from the Moon Estimator convergence proof.

7.4.1 MaxSep Convergence Proof

Theorem 7.4.1 *The MaxSep estimator converges to the slope α as $N \to \infty$.*

Proof As $N \to \infty$, by Theorem 7.1.7, the segment of the upper hull spanning t_{mid} will converge to $\alpha t + \delta$. By a similar argument the segment of the lower hull spanning t_{mid} will converge to $\alpha t - \delta$. these two lines have the minimum possible separation between the two hulls. The MaxSep estimator must therefore select them as having the closest point between the two hulls, and will take as its estimate $\hat{\alpha} = \alpha$. ∎

7.4.2 MaxSep Estimator Convergence Properties

The MaxSep Estimator takes advantage of two-way delay measurements, so it should be expected to out-perform the Moon Estimator. We now obtain a probabilistic bound for the MaxSep Estimator, and conclude by showing that the MaxSep Estimator does indeed converge significantly faster than the Moon Estimator.

We proceed in a similar fashion to before, by considering a hypothetical *anchored estimator* which is guaranteed to give estimates at least as steep as the MaxSep Estimator. We can use its convergence properties to provide a bound on the convergence properties of the MaxSep Estimator.

Consider Figure 7.10, which shows upper and lower bound packet delay measurements $D = \{d_0 \,..\, d_{N-1}\}$ and $Q = \{q_0 \,..\, q_{M-1}\}$ along with their respective hulls. $a_1 t + b_1 + \delta$ is the steepest line passing through the point $(t_{mid}, \alpha t_{mid} + \delta)$ such that $d_i \geq a_1 t + b_1 + \delta$ for all i. Likewise, $a_2 t + b_2 - \delta$ is the steepest line passing through the point $(t_{mid}, \alpha t_{mid} - \delta)$ such that all $q_j \leq a_2 t + b_2 - \delta$. We shall now prove that the steepest of these lines is an upper bound on the convergence of the MaxSep Estimator.

The MaxSep Estimator will choose for its slope estimate a segment from one of the hulls, at the point where the hulls have minimum vertical separation. We need to show that the steeper of the two anchored estimator lines is at least as steep as

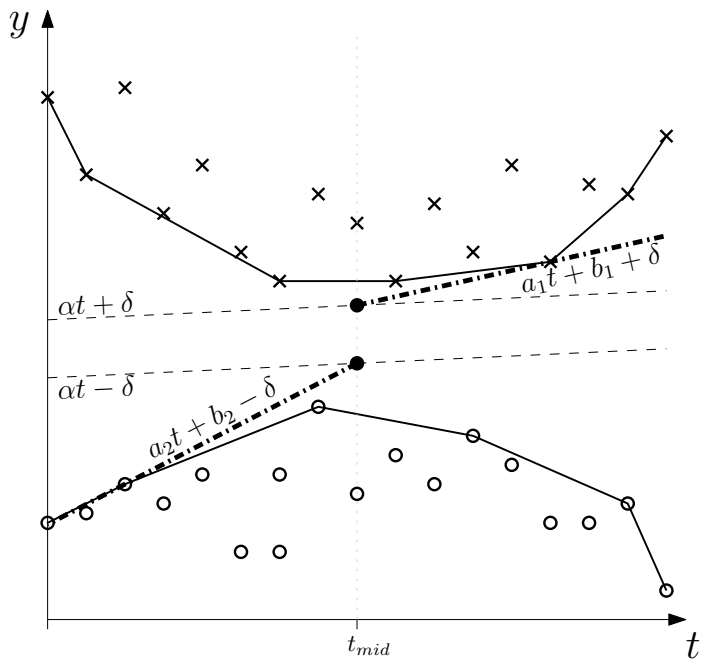


Figure 7.10: The MaxSep Anchored Estimator uses a line crossing $\alpha t + \delta$ at t_{mid} and lying at a tangent to the right half of the top hull, and a line crossing $\alpha t - \delta$ at t_{mid} and lying at a tangent to the left half of the bottom hull. Represented here as a thick dash-dotted lines, one or the other is guaranteed to be at least as steep as the MaxSep Estimator at all times.

any hull segment that the MaxSep estimator may choose. As before, we use $\hat{\alpha}t + \hat{\beta}$ to denote the line passing along the hull segment chosen by the MaxSep estimator.

Lemma 7.4.2 *If the MaxSep Estimator chooses a segment on one hull as its estimate, then its slope is bounded by the slopes of the two segments adjacent to the closest point on the other hull.*

Proof Figure 7.11 shows the situation. The MaxSep estimator finds the point of smallest vertical separation between the two hulls, which by Lemma 7.3.3 happens when they have equal slope. The slope at a hull vertex can take any value bounded by the slope of the two adjacent segments. So those two adjacent segments must also bound the slope of the closest segment on the other hull. ∎

We need to prove that the Anchored Estimate is at least as steep as the MaxSep Estimate for any hull segment that might be chosen by the MaxSep Estimator. We need to consider each end of the two hulls separately, leading to four possible scenarios:

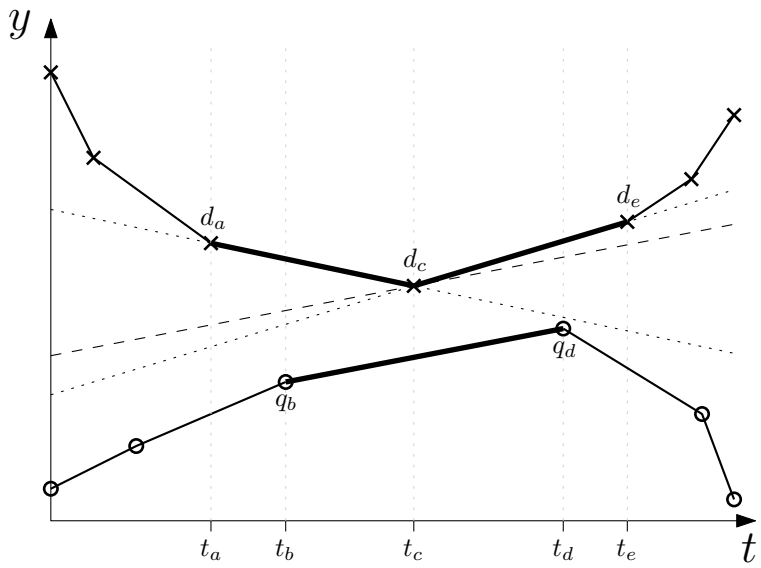


Figure 7.11: The closest vertical distance between the two hulls occurs at t_c, corresponding to vertex d_c on the upper hull. The opposite segment, $\{q_b, q_d\}$ on the lower hull has its slope bounded by segments $\{d_a, d_c\}$ and $\{d_c, d_e\}$ on the upper hull. The dashed line has the same slope as the segment $\{q_b, q_d\}$. If the slope of $\{q_b, q_d\}$ was adjusted such that it was just greater than that of $\{d_c, d_e\}$ then the closest point between the two hulls would snap to t_d. Note that *any* segment on the upper hull that is left of d_c must be less steep than $\{q_b, q_d\}$. Equally, *any* segment on the upper hull that is right of d_c must be steeper than $\{q_b, q_d\}$.

Lemma 7.4.3 *If the MaxSep Estimator chooses a segment on the top hull which is left of, or spanning t_{mid} then $a_1 t + b_1 + \delta$ is steeper than $\hat{\alpha}t + \hat{\beta}$.*

Proof Lemma 7.2.1 showed that $a_1 t + b_1 + \delta$ must be steeper than the segment on the upper hull spanning t_{mid}. It must therefore be steeper than any segment left of t_{mid} on the upper hull. ∎

Lemma 7.4.4 *If the MaxSep Estimator chooses a segment on the top hull which is right of the segment spanning t_{mid} then $a_2 t + b_2 - \delta$ is steeper than $\hat{\alpha}t + \hat{\beta}$.*

Proof See Figure 7.11. By Lemma 7.4.2, the slope of $\hat{\alpha}t + \hat{\beta}$ is bounded by the slopes of the two segments adjacent to the closest point on the lower hull. By Lemma 7.2.1, $a_2 t + b_2 - \delta$ must be at least as steep as the segment on the lower hull spanning t_{mid}, which must in turn be steeper than the two segments bounding the slope of $\hat{\alpha}t + \hat{\beta}$, as they are to the right of t_{mid}. ∎

Lemma 7.4.5 *If the MaxSep Estimator chooses a segment on the lower hull which is left of t_{mid} then $a_1 t + b_1 + \delta$ is steeper than $\hat{\alpha}t + \hat{\beta}$.*

Proof This is a mirror image of Lemma 7.4.4 and may be proven in the same way. ∎

Lemma 7.4.6 *If the MaxSep Estimator chooses a segment on the lower hull which is right of the segment spanning t_{mid} then $a_2 t + b_2 - \delta$ is steeper than $\hat{\alpha}t + \hat{\beta}$.*

Proof This is a mirror image of Lemma 7.4.3 and may be proven in the same way. ∎

Theorem 7.4.7 *If the MaxSep Anchored Estimator, $at+b$ is defined as the steepest of the two lines, $a_1 t + b_1 + \delta$ and $a_2 t + b_2 - \delta$ then $at + b$ must be at least as steep as the MaxSep Estimate, $\hat{\alpha}t + \hat{\beta}$.*

Proof Lemmas 7.4.3, 7.4.4, 7.4.5 and 7.4.6 take account of all possible hull segments that may form the MaxSep Estimate. They show that in all cases, the MaxSep Estimator is bounded above in slope by either a_1 or a_2. If $a = \max(a_1, a_2)$ then *a must* be at least as steep as $\hat{\alpha}$, the MaxSep slope estimate.

This is a key result. As we showed before with the Moon Anchored Estimator, *no matter* what slope the MaxSep Estimate takes, the MaxSep Anchored Estimator will *always* be at least as steep. Again, if we can derive convergence properties for the MaxSep Anchored Estimator then the MaxSep Estimator must converge at least as rapidly as that.

7.4.3 MaxSep Anchored Estimator Convergence Properties

We have shown that any hull segment chosen by the MaxSep Estimator is bounded above in slope by the line $at + b$, which is the Anchored Estimate. Suppose the MaxSep Anchored Estimator has slope error ε_{anc}, so that $a = \alpha + \varepsilon_{anc}$. For that error to be strictly greater than some value, ϕ then all measurements $d_i \in D$ must

lie above the line $(\alpha+\phi)t+b+\delta$, and all measurements $q_j \in Q$ must lie below the line $(\alpha+\phi)t+b-\delta$. This is trivially satisfied for one half of each line; for the other halves we express the slope error probability as:

$$P(\varepsilon_{anc} > \phi) = \prod_{j=0}^{m-1} P(q_j \leq (\alpha+\phi)Tj + b - \delta) \prod_{i=m}^{N-1} P(d_i > (\alpha+\phi)Ti + b + \delta), \tag{7.32}$$

but because $(\alpha+\phi)t+b$ intersects αt at t_{mid}, then $b = -\phi t_{mid}$, so

$$\begin{aligned} P(\varepsilon_{anc} > \phi) &= \prod_{j=0}^{m-1} P(q_j \leq \alpha Tj + \phi(Tj - t_{mid}) - \delta) \\ &\quad \prod_{i=m}^{N-1} P(d_i > \alpha Ti + \phi(Ti - t_{mid}) + \delta) \\ &= \prod_{j=0}^{m-1} P(\alpha Tj - \delta - w_j \leq \alpha Tj + \phi(Tj - t_{mid}) - \delta) \\ &\quad \prod_{i=m}^{N-1} P(\alpha Ti + \delta + w_i > \alpha Ti + \phi(Ti - t_{mid}) + \delta) \\ &= \prod_{j=0}^{m-1} P(w_j > \phi(t_{mid} - Tj)) \prod_{i=m}^{N-1} P(w_i > \phi(Ti - t_{mid})) \end{aligned} \tag{7.33}$$

Noticing that $t_{mid} = \frac{N-1}{2}T$ and $m = \left\lceil \frac{N-1}{2} \right\rceil$ we can simplify the expression to

$$P(\varepsilon_{anc} > \phi) = \prod_{i=0}^{m-1} P(w_i > \phi Ti)^2 \tag{7.34}$$

Converting to the more standard CDF form as we did for the Moon Estimator in equation 7.19 gives,

$$P(\varepsilon_{anc} \leq \phi) = 1 - \prod_{i=0}^{m-1} [1 - P(w_i \leq \phi Ti)]^2 \tag{7.35}$$

7.4.4 MaxSep Estimator Convergence Bound

The MaxSep Anchored Estimate, $at + b$ is *guaranteed* (by Theorem 7.4.7) to be at least as steep as the MaxSep Estimate, $\hat{\alpha}t + \hat{\beta}$. If ε_{MS} is the slope error of the MaxSep Estimator then it must be the case that

$$P(\varepsilon_{MS} \leq \phi) \geq P(\varepsilon_{anc} \leq \phi), \tag{7.36}$$

leading us to the main result of this section, which is a bound on the probability distribution of slope error for the MaxSep Estimator:

$$\boxed{P(\varepsilon_{MS} \leq \phi) \geq 1 - \prod_{i=0}^{m-1} \left[1 - P(w_i \leq \phi T i)\right]^2} \tag{7.37}$$

Compared with the Moon Estimator CDF:

$$P(\varepsilon_{Moon} \leq \phi) \geq 1 - \prod_{i=0}^{m-1} \left[1 - P(w_i \leq \phi T i)\right], \tag{7.38}$$

we see that the MaxSep Estimator can be expected to converge much faster than the Moon Estimator. Given the modest extra overhead of the algorithm, this is a significant improvement. Note also that the convergence rate of both the Moon Estimator and the MaxSep Estimator is *independent* of the minimum network propagation delay, δ.[1]

7.5 Quantifying Estimator error

In this section we look at a way to quantify the convergence properties of the MaxSep Estimator. If we have a suitable model for the distribution of packet delays then we

[1]The same is not true for the offset estimates, since the Moon Estimator's offset estimate is always biased by δ.

can substitute it into equation 7.35 and ask questions about the number of samples required for the algorithm to converge to a required level of accuracy.

Chaudhari et al. [19] model network delays with both Gaussian and Exponential distributions. Elteto and Molnar [37] analyze TCP/IP network delays and propose a technique to learn the distribution, by modelling it with a truncated Gaussian. Sirdey and Maurice [103] note that the Weibull distribution [116] is appropriate, as it well represents the characteristic steep beginning and long tail of typical internet delay histograms. This is supported by Papagiannaki et al. [87] who analyze single-hop network delays and again propose the Weibull distribution as a suitable model. Hernández and Phillips [48] show how to fit a Weibull mixture model to internet packet delay data using an Expectation Maximization technique. They suggest that a mixture of 3 Weibull distributions typically gives a very good fit, though a reasonable approximation is achieved with a single Weibull distribution.

7.5.1 The Weibull distribution

We will use the Weibull distribution primarily because of its flexibility and the fact that it can be fitted to small datasets [33]. Another key advantage is the availability of algorithms for learning the parameters of the distribution online and incrementally. This does limit us to using a single distribution rather than a mixture, but we will show that using a single Weibull distribution will usually cause us to generate close, but conservative estimates of an estimator's error.

The Weibull distribution [116, 23] is a more general form of the Exponential Distribution. It takes two parameters: shape, k and scale, λ. It has the form,

$$f(t;\lambda,k) = \begin{cases} \frac{k}{\lambda}\left(\frac{t}{\lambda}\right)^{k-1} e^{-\left(\frac{t}{\lambda}\right)^{k}} & t \geq 0 \\ 0 & t < 0 \end{cases}, \tag{7.39}$$

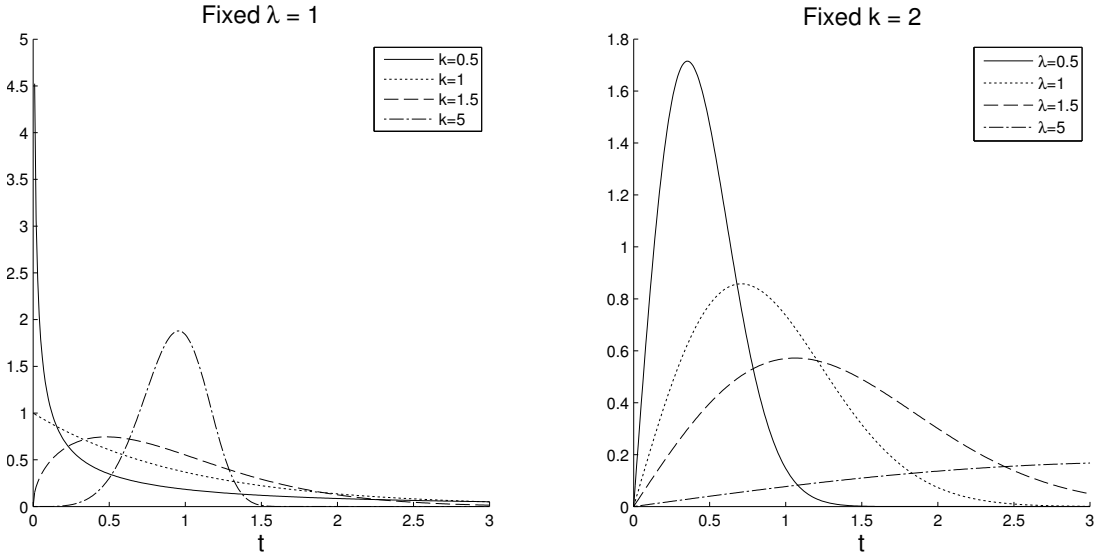


Figure 7.12: The effect of varying the shape (k) and scale (λ) parameters of the Weibull distribution.

with the Cumulative Distribution Function being given by

$$F(t;\lambda,k) = 1 - e^{-\left(\frac{t}{\lambda}\right)^k} \tag{7.40}$$

so that if $w_i \sim f(\lambda, k)$ then

$$P(w_i \leq \phi) = F(\phi;\lambda,k). \tag{7.41}$$

Figure 7.12 shows how the shape of the Weibull distribution varies with k. Techniques for learning the distribution of packet delays online will be discussed shortly.

Recall that the bound on the convergence of the MaxSep Estimator is given by

$$P(\varepsilon_{MS} \leq \phi) \geq 1 - \prod_{i=0}^{m-1} \left[1 - P(w_i \leq \phi Ti)\right]^2 \tag{7.42}$$

where $m = \left\lceil \frac{N-1}{2} \right\rceil$.

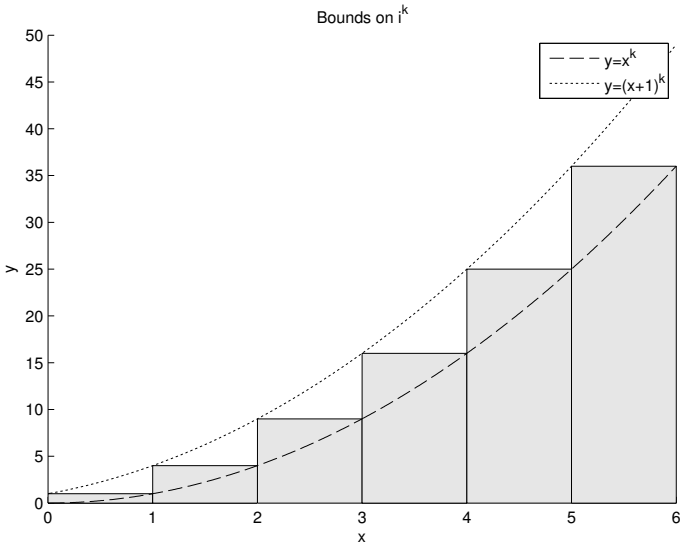


Figure 7.13: The discrete function i^k (shown as a bar plot) is bounded above and below by the continuous functions $y = (x+1)^k$ and $y = x^k$ respectively. In this case, $k = 2$, but the bounds are correct for all $k \geq 0$

Inserting the Weibull CDF gives

$$P(\varepsilon_{MS} \leq \phi) \geq 1 - \prod_{i=0}^{m-1} e^{-2\left(\frac{\phi T i}{\lambda}\right)^k} \tag{7.43}$$

In log form this becomes

$$\ln\left[P(\varepsilon_{MS} > \phi)\right] \geq -2 \sum_{i=0}^{m-1} \left(\frac{\phi T i}{\lambda}\right)^k \tag{7.44}$$

$$= -2 \left(\frac{\phi T}{\lambda}\right)^k \sum_{i=0}^{m-1} i^k \tag{7.45}$$

Finding a bound on the true distribution

The summation term in equation 7.44 has no closed form solution. Instead we must resort to finding bounds on the true result. Consider the function

$$g(N) = \sum_{i=0}^{N-1} i^k \tag{7.46}$$

Figure 7.13 shows that the discrete function i^k is bounded above and below by the continuous functions $y = (x+1)^k$ and $y = x^k$. Thus we may obtain the following bounds on the function $g(N)$,

$$\int_0^{N-1} x^k dx \quad < \quad g(N) \quad < \quad \int_0^{N-1} (x+1)^k dx \tag{7.47}$$

$$\left[\frac{x^{k+1}}{k+1}\right]_0^{N-1} \quad < \quad g(N) \quad < \quad \left[\frac{(x+1)^{k+1}}{k+1}\right]_0^{N-1} \tag{7.48}$$

$$\frac{(N-1)^{k+1}}{k+1} \quad < \quad g(N) \quad < \quad \frac{N^{k+1}}{k+1} \tag{7.49}$$

As $N \to \infty$, the bounds converge, because

$$\lim_{N\to\infty} \frac{(N-1)^{k+1}}{k+1} = \frac{N^{k+1}}{k+1} \tag{7.50}$$

and in practice the bounds are tight for $N >> 1$.

Applying this result to equation 7.44 gives

$$\ln\left[P(\varepsilon_{MS} > \phi)\right] \geq -2\left(\frac{\phi T}{\lambda}\right)^k \frac{m^{k+1}}{k+1} \tag{7.51}$$

Approximating $m \approx (N-1)/2$ and rearranging gives

$$\boxed{\ln\left[P(\varepsilon_{MS} > \phi)\right] \geq -\left(\frac{\phi}{\psi\lambda}\right)^k, \quad \text{where} \quad \psi = \frac{2}{T(N-1)}\left(\frac{k+1}{N-1}\right)^{\frac{1}{k}}} \tag{7.52}$$

7.5.2 Quantitative estimates

Differentiating equation 7.52 allows us to recover the PDF over slope error for the MaxSep Anchored Estimator,

$$\begin{aligned} p(\varepsilon_{anc}) &= -\frac{d}{d\phi} P(\varepsilon_{anc} > \phi) \\ &= \frac{k}{\psi\lambda} \left(\frac{\varepsilon_{anc}}{\psi\lambda} \right)^{k-1} e^{-\left(\frac{\varepsilon_{anc}}{\psi\lambda}\right)^k} \end{aligned} \tag{7.53}$$

which is simply a form of the Weibull distribution, with $p(\varepsilon_{anc}) = f(\varepsilon_{anc}; \psi\lambda, k)$ - ie. a scaled version of the original distribution. This is a pleasing result, especially given the approximations used in the derivation.

A Weibull distribution has mean and variance:

$$\mathrm{E}(t) = \lambda\Gamma\left(1 + \frac{1}{k}\right) \tag{7.54}$$

$$\mathrm{var}(t) = \lambda^2\Gamma\left(1 + \frac{2}{k}\right) - \mathrm{E}(t)^2 \,. \tag{7.55}$$

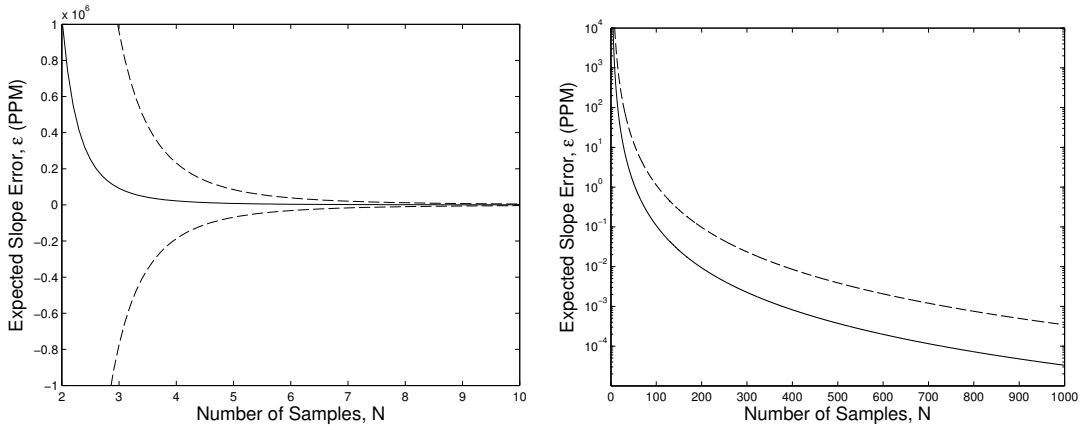
Γ is the Gamma function, which has no closed form solution, but there are many fast and accurate approximation algorithms such as those by Cody [22]. Using these we can calculate the expected error, given the number of samples, N.

From the previous inequalities on the CDFs for the MaxSep Estimator and the Anchored Estimator, we know that the error of the MaxSep Estimator will not be greater than the error of the Anchored Estimator. We can thus state that,

$$\mathrm{E}(\varepsilon_{MS}) \leq \psi\lambda\Gamma\left(1 + \frac{1}{k}\right) \tag{7.56}$$

$$\mathrm{var}(\varepsilon_{MS}) \leq (\psi\lambda)^2 \left[\Gamma\left(1 + \frac{2}{k}\right) - \left[\Gamma\left(1 + \frac{1}{k}\right) \right]^2 \right] . \tag{7.57}$$

Sirdey and Maurice [103] suggest suitable Weibull distribution parameters for a



(a) **(b)**

Figure 7.14: Expected slope error of the MaxSep Estimator for 50Hz sampling and Weibull parameters $k = 0.4$ and $\lambda = 1.35 \times 10^{-3}$. The dotted lines are 3σ bounds. Note the rapid convergence. The figure on the right shows the same data on a log scale.

public internet setting to be $k = 0.4$ and $\lambda = 1.35 \times 10^{-3}$. Figure 7.14 shows the expected error with a sample rate of $50Hz$, along with 3σ bounds calculated from Equation 7.57.

Following the same derivation for the Moon Estimator, we find that

$$\mathrm{E}(\varepsilon_{Moon}) \leq 2^{\frac{1}{k}} \psi\lambda\Gamma\left(1+\frac{1}{k}\right) \tag{7.58}$$

$$\mathrm{var}(\varepsilon_{Moon}) \leq 2^{\frac{2}{k}} (\psi\lambda)^2 \left[\Gamma\left(1+\frac{2}{k}\right) - \left[\Gamma\left(1+\frac{1}{k}\right)\right]^2\right]. \tag{7.59}$$

Note that these are derived from the anchored estimators, and as such cannot be used to compare completely fairly between the Moon and MaxSep estimators, though they do give a good ball-park for the performance of each.

Determining the number of samples required for a given accuracy

Perhaps more usefully, we may also derive an expression for estimating the minimum number of samples required to achieve a particular slope error ε with a given confidence c. The confidence value lies in the range $[0, 1]$, so that $c = 0.99$ corresponds to

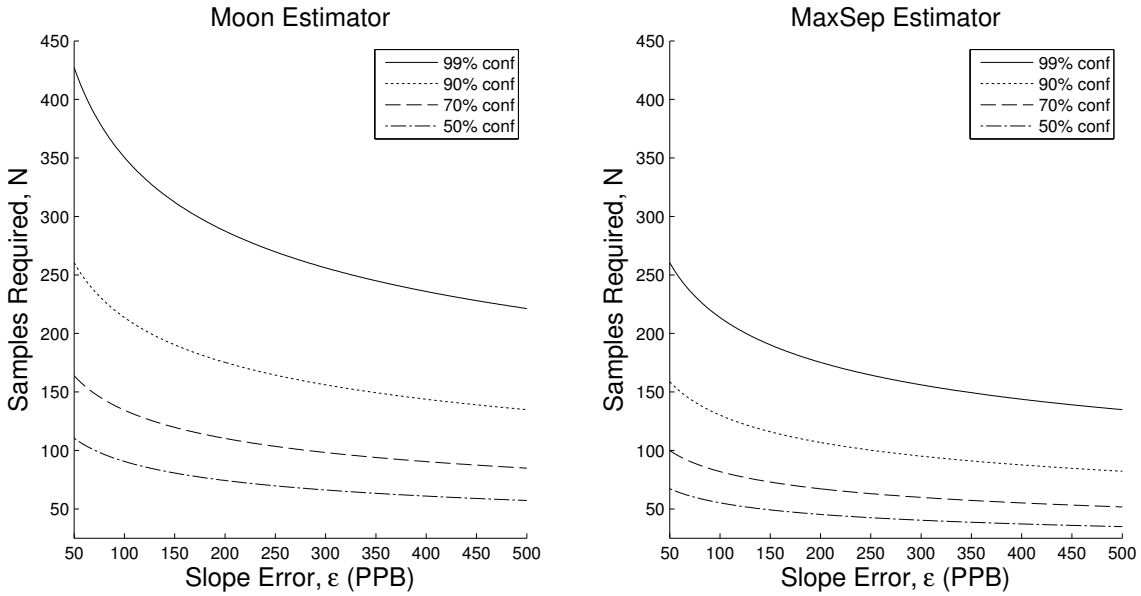


Figure 7.15: Required number of samples to achieve a given level of slope accuracy, with $k = 0.4$ and $\lambda = 1.35 \times 10^{-3}$ and a sampling frequency of 50 Hz. Various confidence levels are shown. The Moon Estimator requires a factor of $2^{\frac{1}{k+1}}$ as many samples as the MaxSep Estimator. Errors are given in Parts Per Billion (PPB).

a 99% confidence bound. Rearranging equation 7.52,

$$\boxed{N_{MS} \geq \left[-(k+1) \left(\frac{2\lambda}{\varepsilon T} \right)^{k} \ln(1-c) \right]^{\frac{1}{k+1}} + 1} \tag{7.60}$$

As a comparison point, a similar expression can be derived for the Moon Estimator:

$$N_{Moon} \geq 2^{\frac{1}{k+1}} \left[-(k+1) \left(\frac{2\lambda}{\varepsilon T} \right)^{k} \ln(1-c) \right]^{\frac{1}{k+1}} + 1 \tag{7.61}$$

Sample plots of these function are shown in Figure 7.15. Note that reducing the required confidence by only a small amount can drastically reduce the required number of samples.

Offset error

Let ε be the MaxSep Estimator's slope error after N samples and let $\varepsilon_{sep} = (\hat{\beta}_1 - \hat{\beta}_2) - 2\delta$ be the separation error. Because of the necessity to touch at least one data point, the upper line of maximum separation, $\hat{\alpha}t + \hat{\beta}_1$ is constrained to lie above the line $\alpha t + \delta$ at some point in the interval $[t_0, t_{N-1}]$. The lower line of maximum separation is similarly constrained to have some part below $\alpha t - \delta$ in the same interval.

The offset error ε_τ after N samples is therefore bounded above and below, such that

$$\begin{cases} -\frac{\varepsilon_{sep}}{2} \leq \varepsilon_\tau \leq \varepsilon TN + \frac{\varepsilon_{sep}}{2} & \varepsilon \geq 0 \\ \varepsilon TN - \frac{\varepsilon_{sep}}{2} \leq \varepsilon_\tau \leq \frac{\varepsilon_{sep}}{2} & \varepsilon < 0 \end{cases} \tag{7.62}$$

Thus the expected value of the offset error is $E[\varepsilon_\tau] = \varepsilon TN/2$. If we use a 99% confidence upper bound on ε then the expression for ε_τ is an approximation of an upper bound on the true offset error. Unfortunately we cannot make a stronger statement than that, but in practice we find it to be a sound estimate.

Figure 7.16 shows slope and offset errors for the MaxSep Estimator when applied to synthetic data for two clocks with constant relative skew. We caused the Estimator to be reset every 50 samples, so that its initialization behaviour can be studied. The equations derived above are used to provide 99% confidence bounds for the plots.

7.5.3 Improving initial convergence

In practice the MaxSep Estimator converges rapidly, but the slope estimate can be unstable for the first few samples. If the packet delay distribution is known a priori then the Maximum Likelihood Estimator from section 6.8.3 may be used initially (while it is still tractable) and the MaxSep Estimate switched in once it has stabilized.

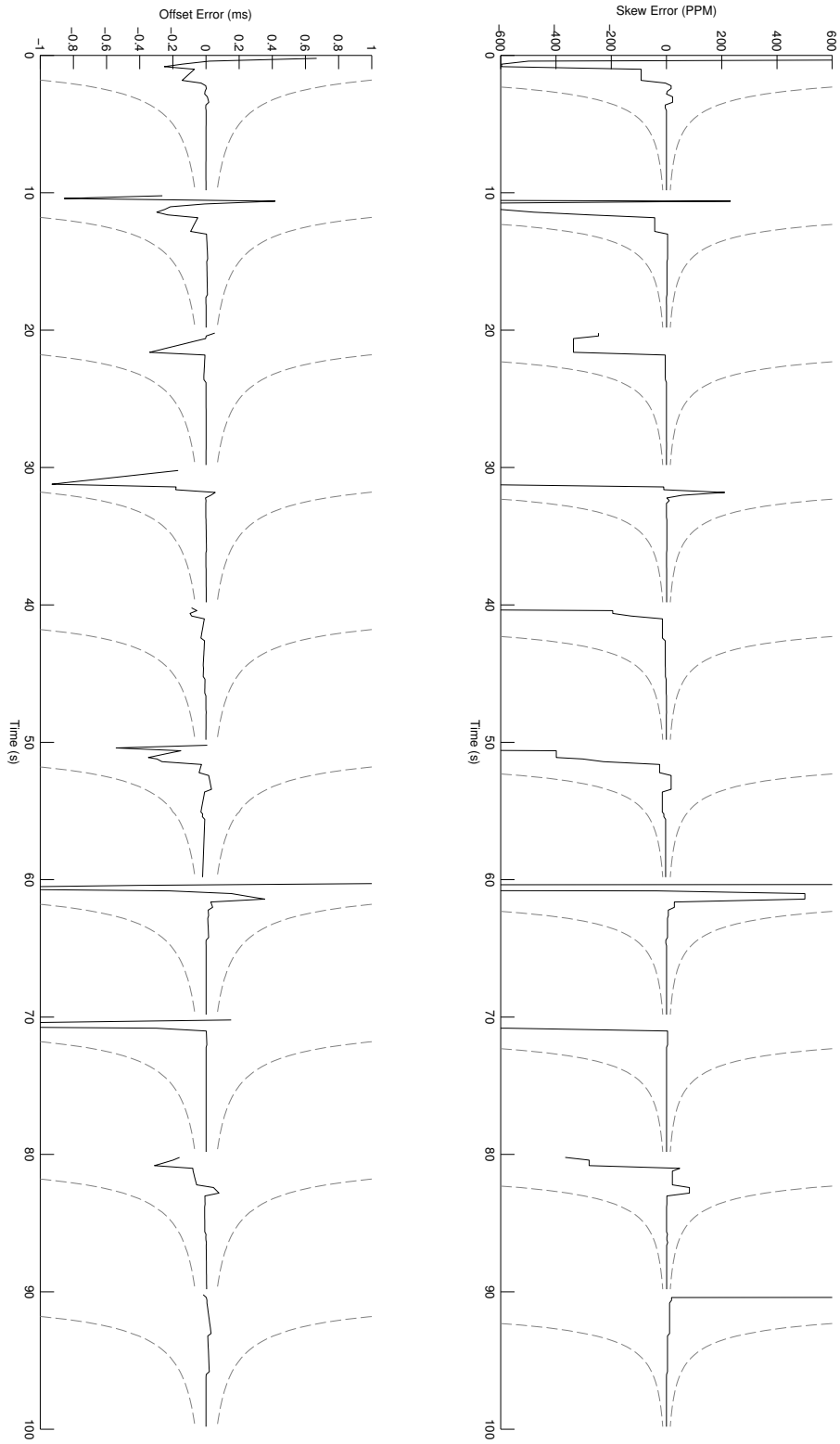


Figure 7.16: Slope and Offset Error plots for the MaxSep Estimator when applied to synthetic data with Weibull Parameters $\lambda = 1.35e-3$ and $k = 0.4$ at $5Hz$. The filter is reset every 50 samples in order to show many convergence cycles. The dashed lines are the 99% confidence bounds.

It may be a suitable course of action if the filter needs to be reset (ie. in the event of a clock upset) and the packet delay distribution can be assumed to be unchanged. The approach eventually becomes intractable because to optimize the estimator's parameters requires iterating over all packet delay data gathered so far, making it $O(N^2)$ if the parameters are updated at every time step. Since clock synchronization is a low-level function, it should only use a small amount of CPU time for updates, so the MLE approach is liable to become too expensive quite rapidly.

Failing that, rapid initial convergence can be achieved by using a faster sample rate for the first few seconds only. This maximizes the probability of obtaining low-latency measurements, which are critical at the start, as the slopes of the short hull segments are not well conditioned.

7.6 Convergence results

In order to evaluate our probabilistic analysis of the convergence properties of the Moon estimator and the MaxSep estimator, we performed experiments with synthetic data. Concretely, we produced a sequence of synthetic network delay measurements, drawn from a Weibull distribution with parameters $\lambda = 1.35 \times 10^{-3}$ and $k = 0.4$, as suggested by Sirdey and Maurice [103]. The minimum network delay was chosen as $\delta = 5$ms, and the sample rate was 10Hz.

3000 different network delay sequences were produced, with random skew and offset between clocks, of up to 1ms/s and 200ms respectively (though we have already shown that these have no effect on the magnitude of the error). Both the Moon and the MaxSep estimators were applied to each sequence, and the mean errors at each time step were found. The results are shown in Figure 7.17, along with the expected values of skew computed using Equations 7.56 and 7.58.

In offset estimation, the Moon estimator suffers from a constant DC error equal to

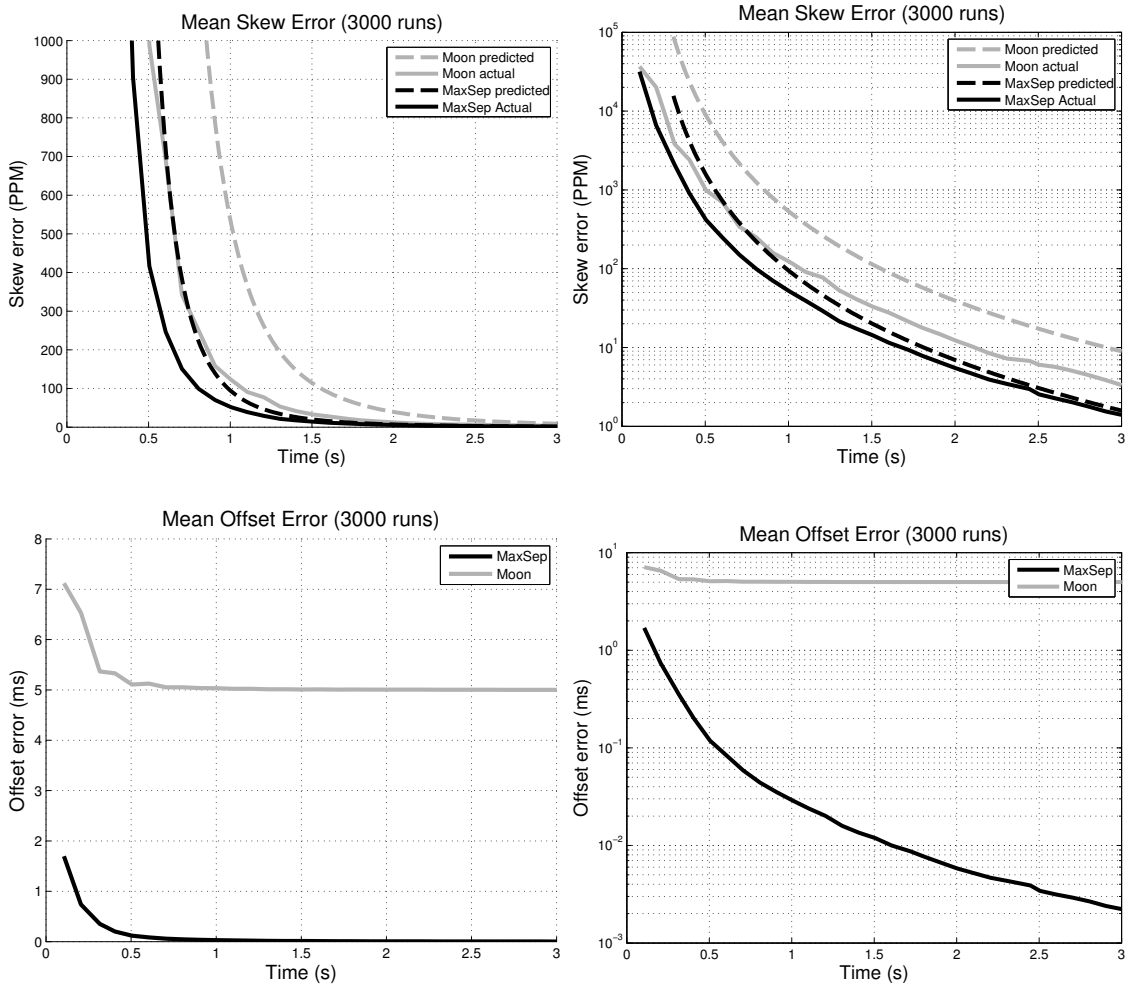


Figure 7.17: Synthetic data experiments to compare the performance of the Moon and MaxSep estimators. Packet delay data was generated from a Weibull distribution with parameters $\lambda = 1.35 \times 10^{-3}$ and $k = 0.4$ at a rate of 10Hz, and fed through both estimators. The experiment was repeated 3000 times on different data each time, and these figures show the mean errors over all experiments. For the skew plots, the dotted lines show the expected error for each estimator.

the minimum network delay, δ. The MaxSep estimator does not suffer from this, so is clearly superior. In skew estimation the MaxSep estimator has a faster convergence rate than the Moon estimator.

The expected skew errors are shown as dotted lines. These are computed using the anchored estimators, which provide an upper bound on the performance of the actual estimators. This explains why both estimators appear to outperform their expected errors.

7.7 Chapter Summary

We have now presented new convergence proofs for two existing clock synchronization methods due to Moon et al. [73] and Sirdey and Maurice [103] - the 'Moon Estimator' and the 'MaxSep Estimator' respectively. We showed how the linear programming approach of Sirdey and Maurice has a dual formulation involving convex hulls and that the MaxSep objective function is maximized simply by finding the point of minimum separation between the convex hulls of the upper and lower packet delay data.

By transforming each method into its dual formulation, a probabilistic upper bound on the convergence characteristics of each method was provided, allowing for a direct comparison between the two. The MaxSep Estimator was found to have significantly improved performance over the Moon Estimator, due to its use of two-way delay data. We also showed that convex hulls applied to network timing data require only a small number of segments to represent them, even when the number of delay measurements they enclose is very large. This is significant because it shows that convex hull based synchronization approaches can be very memory efficient whilst still being able to make optimal estimates involving all previous timing data.

These insights will lead us directly to the creation of the efficient clock synchronization (a.k.a data time stamping) algorithm which we seek in the next chapter.

Chapter 8

An Efficient Algorithm for Clock Synchronization

In this Chapter we build on previous chapters and describe a new efficient clock synchronization (data time-stamping) algorithm for solving the MaxSep objective function. We call it *TICSync*, for *Timestamp-based Incremental Clock Synchronization*. The algorithm is incremental, with each update having $O(1)$ amortized cost, making it highly tractable for on-line use. This is a significant improvement over the algorithm suggested by Sirdey and Maurice [103], since their linear programming algorithm is $O(N)$ for each incremental update if an estimate is required after every new sample and thus $O(N^2)$ in total.

Using the convergence results of Chapter 7 and by choosing an appropriate model for the distribution of packet delay measurements (which can also be learnt online incrementally), we can use the methods developed in Section 7.5 to obtain quantitative estimates (with confidence bounds) of TICSync's current error.

We then extend the basic TICSync algorithm to work in regimes where the clocks do not behave in the manner expected by the linear skew model. It is reasonable to question the regularity of clock upsets. The quartz oscillators of a PC clock are

low-cost items that are sensitive to changes in temperature. Or if a computer is running NTP and has not yet converged, there can be large swings in frequency. Reset events may be caused by a computer using the SNTP protocol to synchronize its clock, by user intervention, or the running of a rogue program which is able to make clock adjustments. Being robust to frequency changes and reset events is vital to accurately tracking the current time at the server's clock.

The Upset Detection scheme runs with $O(F)$ incremental updates, where F is the size of a Bank of Estimators and is usually less than 10. This is significantly more efficient than other upset detection algorithms that we are aware of and crucially is able to detect upsets in an online setting in near-optimal time. We call the improved algorithm TICSync+.

8.0.1 A Convex-Hull based Algorithm

We begin with a simple observation about the linear programming approach to solving the MaxSep Objective Function proposed by Sirdey and Maurice:

$$\begin{aligned} \max_{\hat{\alpha},\hat{\beta}_1,\hat{\beta}_2} \quad & \hat{\beta}_1 - \hat{\beta}_2 \\ \text{s.t.} \quad & \hat{\alpha}t_i + \hat{\beta}_1 \leq d_i, \qquad i = 0\,..\,N-1 \\ & \hat{\alpha}t'_j + \hat{\beta}_2 \geq q_j, \qquad j = 0\,..\,M-1 \end{aligned} \tag{8.1}$$

Sirdey and Maurice use a rapid 50Hz sample rate in their synchronization experiments. They suggest running the algorithm over 10 minute windows of data, over which the error due to clock drift is negligible. For a 10 minute run, the batch linear-programming algorithm must process 60,000 constraints; many of them far from the feasible region and therefore never active.

Our initial observation is that the algorithm's efficiency would be greatly improved

by using as constraints only those points which appear as vertices on their respective convex hulls. In our own experiments, convex hulls applied to timing data rarely exceed 20 segments, independent of the number of samples; the segments just tend to get longer (this statement is justified by the result in Section 7.1.2). The hulls could be maintained incrementally and then batch linear programming updates would be cheap.

In this section, we demonstrate how the algorithm can be even further improved such that the whole optimization may be performed incrementally. To motivate our new TICSync Estimator for solving the MaxSep Objective function, we have already shown (Section 7.3) that maximizing the objective function can be achieved by finding the point of minimum vertical separation between the upper and lower convex hulls of the two-way timing data.

The lower convex hull for a time ordered set of points,

$$V = \{(t_0, v_0), (t_1, v_1), \ldots, (t_{N-1}, v_{N-1})\}, \tag{8.2}$$

can be computed with the simple $O(N)$ package wrapping algorithm given by Zhang et al. [123]. It is reproduced here as Algorithm 2 (Lower-Convex-Hull). By swapping inequalities, the algorithm may be used to compute an upper convex hull. Note that the cost of adding each new point is $O(1)$ amortized. The function 'line(v_a, v_b)' fits a line through the points v_a and v_b.

We can now describe the basic TICSync algorithm to efficiently solve the MaxSep Objective function. There are only a few simple steps. When a new packet delay measurement arrives, it is added incrementally to the appropriate convex hull (upper or lower). Then the procedure Find-Closest-Hull-Vertices is called to find the instant, t_* of minimum vertical distance between the two hulls. The slope of the hulls at t_* gives the slope of the two Maximum Separation lines. Finally the offsets for

Algorithm 2 Lower Convex Hull Algorithm

```
procedure LOWER-CONVEX-HULL({v_0 .. v_{N-1}})
    stack ← {}
    stack.push(v_0)
    stack.push(v_1)
    for i = 2 to N − 1 do
        if v_i above line(stack.top, stack.penultimate) then
            stack.push(v_i)
        else
            while stack.size > 1 and ...
              ... v_i not above line(stack.top, stack.penultimate) do
                stack.pop
            end while
            stack.push(v_i)
        end if
    end for
    return stack
end procedure
```

the Maximum Separation lines are computed such that they contact their respective hulls at t_*.

The procedure FIND-CLOSEST-HULL-VERTICES is easy to implement. Starting from the left hand side and sweeping right, it evaluates the distance between the hulls for every vertex encountered. As soon as the distance begins to increase, then the algorithm may bail out, because the vertical distance between the two hulls is a convex function (Lemma 7.3.1). The algorithm will typically return a vertex index on one hull and a pair of vertex indices on the other hull, corresponding to a segment. If two segments on the upper and lower hull are parallel then the lines of maximum separation will lie through both segments, as is the case in Figure 7.7. A final case is when the lines pass through only a single hull vertex each. In this example the objective function has a flat spot where a whole range of slopes will achieve maximum separation. In that case it is sensible to choose a slope mid-way through the range.

FIND-CLOSEST-HULL-VERTICES is already efficient enough to be run at every time step, since the convex hulls tend to have few segments. Nonetheless it may be

further improved by noticing that a) the point of minimum distance between two hulls will *only* change when a new measurement falls between the existing Lines of Maximum Separation and b) it will only ever move to the right.

The full update for a new upper bound delay measurement is given by the following steps:

- Add the point to the upper convex hull
- If the point lies below $\hat{\alpha}t + \hat{\beta}_1$ then
  - Call FIND-CLOSEST-HULL-VERTICES to search *right* from the previous point of minimum hull distance.
  - Recompute the lines of maximum separation to pass through the new closest hull vertices or segments.

The update for lower bound measurements is similar. The cost of an update is O(1) amortized: usually there is no work to do beyond adding a new hull segment, but occasionally the maximum separation lines will need to be recomputed. As the algorithm converges to the correct line, the regularity with which a datum falls between the Maximum Separation Lines will decrease to zero. Once the algorithm has converged, the update cost is minimal.

8.1 Learning the Weibull Distribution Online

In order to make good predictions of error or required sample counts, it is important that the network delay distribution is well characterized. In practice we need to learn the distribution online as data is accumulated.

The input data is derived from the packet round trip times. We need a way to deal with the minimum network delay portion of the round trip times, as this is not modelled by the two-parameter Weibull distribution. One way is to adjust the round

trip time (RTT) data by a current TICSync estimate of the minimum network delay, δ. The other way is to add a third parameter representing position to the Weibull distribution and learn all three directly from the data.

The 3-parameter Weibull distribution is written as

$$p(t;\lambda,k,\rho) = \begin{cases} \frac{k}{\lambda}\left(\frac{t-\rho}{\lambda}\right)^{k-1} e^{-\left(\frac{t-\rho}{\lambda}\right)^{k}} & t \geq \rho \\ 0 & t < \rho \end{cases}, \tag{8.3}$$

where ρ is the position parameter.

8.1.1 Weibull Parameter Estimators

Estimators for the 2-parameter and 3-parameter Weibull Distribution are described by Cohen and Whitten [23] and Ghosh [42] amongst others. The Maximum Likelihood Estimator (MLE) is commonly used and is generally regarded to give the best result (under the right conditions). Let $\{x_0 \ldots x_n\}$ be a set of samples drawn from a Weibull distribution. The likelihood function for the 3-parameter case is

$$L(x_0 \ldots x_n;\lambda,k,\rho) = \left(\frac{k}{\lambda^k}\right)^n \prod_{i=1}^{n}(x_i-\rho)^{k-1} \quad \exp\left[-\sum_{i=1}^{n}\left(\frac{x_i-\rho}{\lambda}\right)^k\right] \tag{8.4}$$

The Maximum Likelihood Estimator is an iterative algorithm, using successive estimates to refine the Weibull parameters of a given delay sequence. One disadvantage of the approach is that it requires all of the data to be kept and iterated over whenever a new sample arrives, making an incremental parameter fitting algorithm at least $O(N)$ per update. Perhaps more significant is that there are no valid solutions for some values of the shape parameter. There are known numerical issues for values of $k < 2$ and worse, when $k < 1$ the likelihood function becomes infinite. Cohen and Whitten [24] suggest a series of Modified Maximum Likelihood Estimators in an attempt to overcome the issues.

More interesting in the context of our work is the Modified Moment Estimator (MME), also due to Cohen and Whitten [24]. It reportedly performs better than the MLE for smaller sample sizes, is applicable for any value of the shape parameter k, and remarkably requires as input only three population statistics: the mean, the variance and the first order statistic (minimum value). Appendix F describes the steps of the algorithm.

The MME algorithm is easily made to run online, with $O(1)$ updates. The variance and mean can be incrementally updated using the method of Knuth [56] (See Appendix G) and maintaining the minimum is trivial. The MME does use an iterative root-finding step to determine k, but from one sample to the next the shape parameter is unlikely to change by much, so a small number of iterations are usually sufficient.

8.1.2 Using the Modified Moment Estimator on Real Data

The Modified Moment Estimator is a remarkable algorithm, but it relies fundamentally on the assumption that the sample values really do come from a Weibull distribution. Figure 8.1 shows some real network delay histograms from a week-long experiment we carried out on our local Gigabit network. The packet delay distribution is clearly multi-modal, so the Weibull model is not strictly correct. But recall that our expression for the TICSync estimator error in Equation 7.42 uses the CDF of the distribution, rather than the PDF. In fact, the Weibull model follows the CDF of the raw data rather well.

Conservative Estimates

If the initial part of the Weibull CDF is too steep, or leads the true CDF then it will result in overconfident TICSync error estimates. Conversely, if the Weibull CDF lags the true CDF or begins at too shallow a slope then it will result in conservative

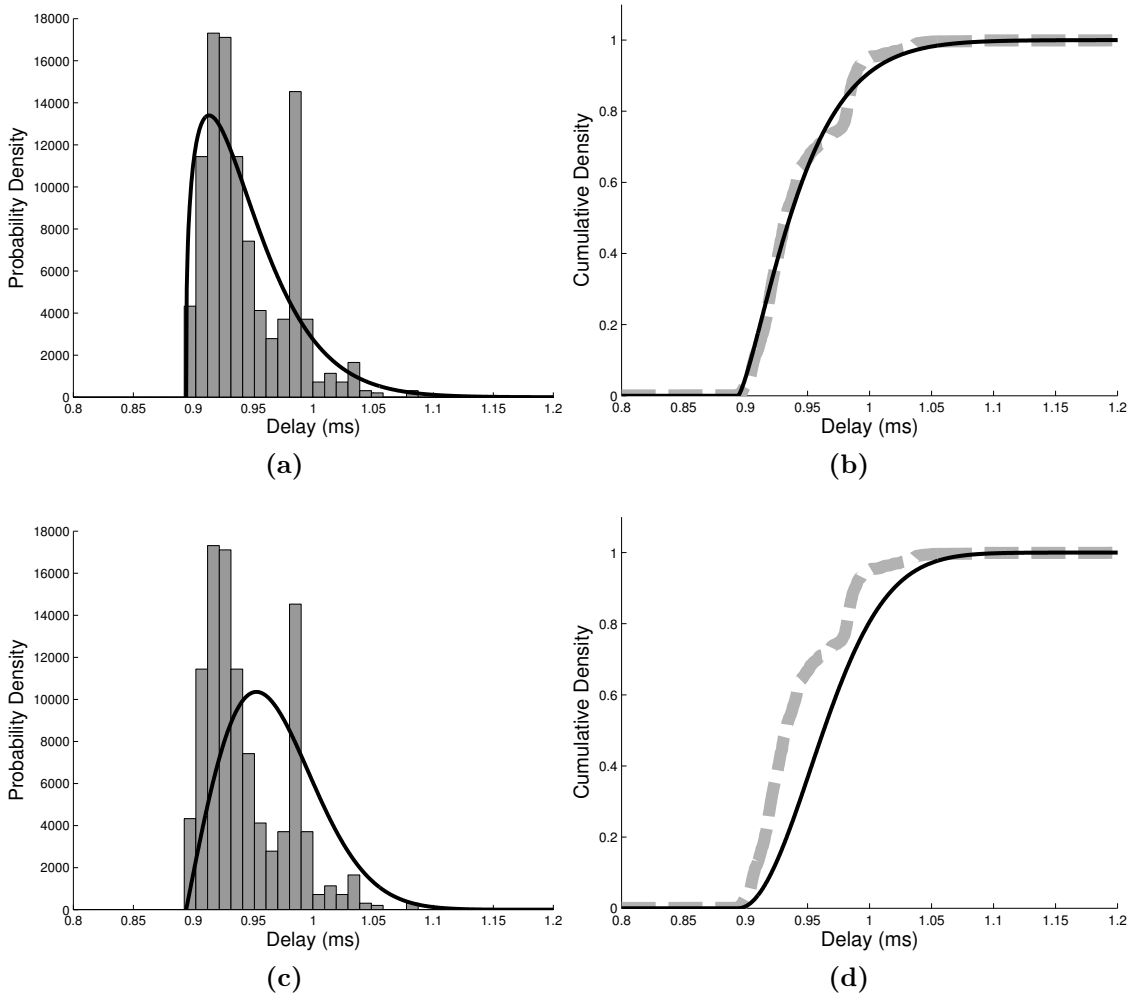


Figure 8.1: A packet delay measurement experiment was performed over a fast Gigabit network between two standard PCs, with a message rate of 2Hz. (a) shows a histogram of 1000 delay measurements, along with the Weibull Distribution fitted by the Modified Moment Estimator (parameters: $\lambda = 5.4 \times 10^{-5}, k = 1.33, \rho = 0.89$ms). It is clear that the data is not really drawn from a Weibull distribution. However, our error estimates make use of the Cumulative Distribution rather than the PDF. (b) shows the CDF of the data as a thick dashed line, with the CDF of the Weibull distribution overlaid. The similarity here is much greater. (c) shows the same histogram, but this time the MME was forced to choose a shape parameter value of $k > 2$. The front end of the distribution is less steep, leading to a conservative estimate of the CDF (d).

error estimates. The latter situation is clearly preferable, so on that basis we place a restriction on the minimum value of the shape parameter for our Weibull distributions, enforcing that $k > 2$. The decision is not arbitrary, as 2 is the smallest value for which the initial slope of the Weibull function is 0. It helps to ensure that the sharp front edge of the true histogram is only softly reproduced by the Weibull distribution. Figure 8.1 shows the result of applying the restriction. We find that the approximation works very well throughout our week-long dataset, giving a Weibull CDF which is close to the true data, but usually conservative. To reiterate, it is not necessary that the Weibull Distribution exactly fits the true network delay distribution. The important thing is that the Weibull CDF is conservative in comparison to the true CDF. That way, the error estimates that the following TICSync+ algorithm relies upon will always be conservative.

Outlier Rejection

One final component is required to make Weibull parameter fitting a success. We find it necessary to perform simple outlier detection on the incoming data before fitting the distribution. Occasionally network packets can get lost or delayed for a considerable period. Just one erroneous measurement can greatly bias the mean and variance statistics used by the MME. We apply a standard χ^2 gating procedure based on the previous mean and variance estimates to throw out measurements which are very improbable. We have found this simple measure to be highly effective.

8.1.3 Sliding Window Weibull Estimation

In order to track changes in network properties over time, we suggest running the MME at occasional intervals of around 200 samples or 5 minutes (whichever is smaller). That gives a good compromise between the quality of Weibull estimates and the phase lag of the MME.

A more expensive alternative is to run the MME incrementally over a sliding window. It would require the ability to *remove* samples from the running mean, variance and first order statistics as they drop off the end of the window. Using a ring-buffer of previous data makes this trivial for the mean and variance values. A running minimum in amortized $O(1)$ time is a little harder to achieve, but can be done by maintaining a list of ascending minima throughout the sliding window [118].

A simple way of initializing the distribution parameters quickly would be to use a very rapid sample rate for a few seconds when the MME is first started up. Empirically, we find on synthetic data that the estimated distribution rapidly converges towards the correct shape.

8.2 Upset Detection

None of the existing work appears to make full use of two-way timing data for upset detection. We present efficient and simple online algorithms for rapid (sometimes instantaneous) detection of upsets.

We first reiterate that the offset between two clocks is bounded by the two-way packet delay measurements. Thus the offset estimate need never be in error by more than the most recent Round Trip Time.

8.2.1 Large Reset Detection

Detection of large resets greater in magnitude than the current closest vertical distance between the two hulls is trivial. Such resets will cause an immediate intersection of the two hulls, requiring the filter to be reset.

Because Large Reset detection is instant, hiccups of arbitrary duration are handled too. Figure 8.2 shows the filter output and slope and offset error plots for a synthetic case, showing the instantaneous reaction of the filter to reset events.

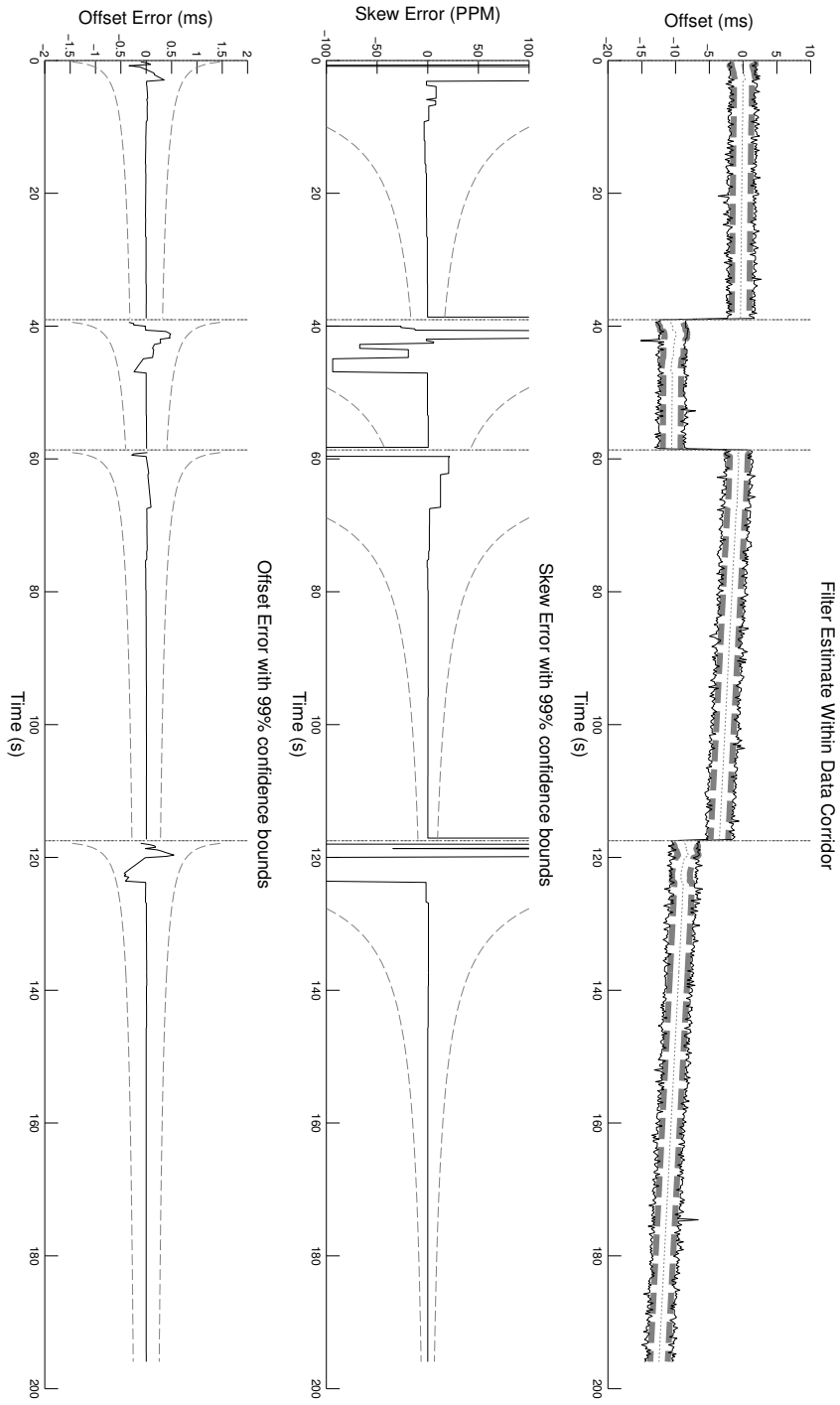


Figure 8.2: Filter response to a series of reset events, using synthetic data at 5Hz. The data were produced by measuring a series of real packet round trip times and randomizing their order, before applying artificial skew and offset events. The thick dashed lines show the estimates from the lines of maximum separation, with the fine dotted line being the final offset estimate. The middle plot shows that the slope error remains within the 99% confidence interval predicted using a Weibull distribution fitted to the data. The bottom plot shows that the offset error rapidly reduces to well below 50µs. Indeed, by the end of the plot, the achieved offset error is around 0.5µs.

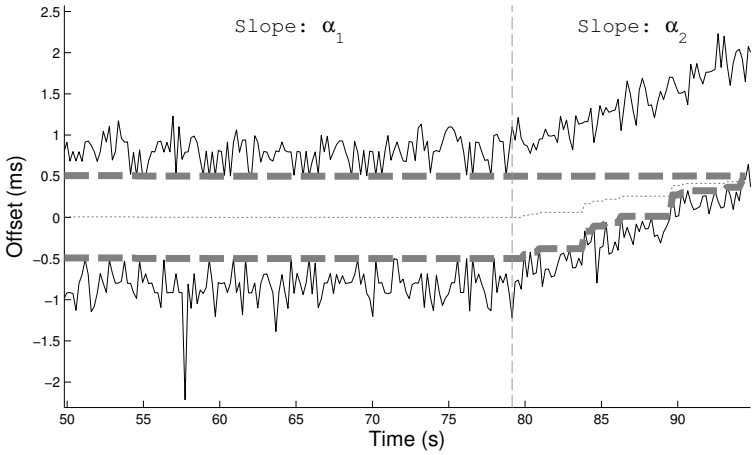


Figure 8.3: Response of a previously converged filter to a skew change. The thick lines show the estimates from the lines of maximum separation. If the slope changes from α_1 to α_2 then the time to the meeting of the lines of maximum separation is $\Delta \approx 2\delta / |\alpha_2 - \alpha_1|$.

8.2.2 Skew Change Detection - A Naïve Approach

A similar mechanism might be used to detect changes in skew. We will analyze the response to a skew change to see whether it results in satisfactory detection.

Figure 8.3 shows the situation for a filter that has converged on an initial slope. The initial slope is α_1, the new slope is α_2 and in this case $\alpha_2 > \alpha_1$. After the skew change, the minimum distance between the two hulls will decrease linearly until the hulls intersect, or the maximum separation lines meet.

The time until intersection is a function of both α_1 and α_2, and is the time taken for the offset to vary by 2δ. The detection delay is given by

$$\Delta = \frac{2\delta}{|\alpha_2 - \alpha_1|} \tag{8.5}$$

This should be regarded as a best-case scenario. If the algorithm has not previously converged to α_1 then it might run for much longer before intersection occurs. Figure 8.4 demonstrates how the basic TICSync Estimator can maintain an incorrect but feasible solution for much longer, if the previous slope was only seen for a finite time.

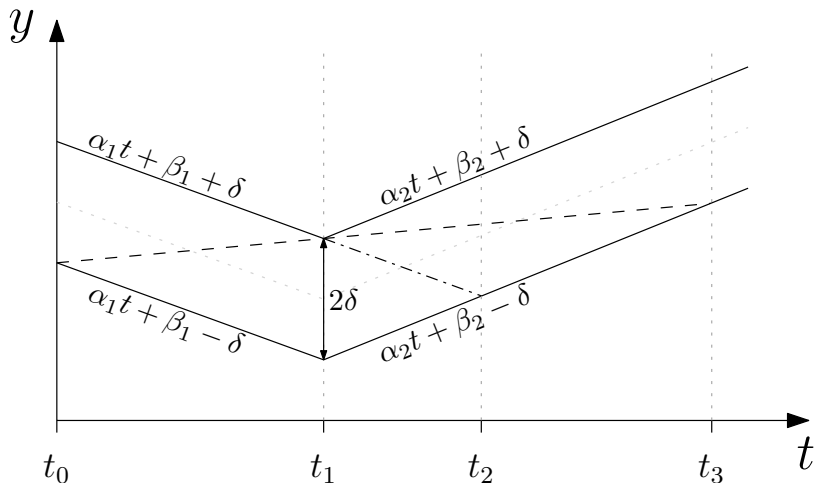


Figure 8.4: The bounds corridor is modelled by the two solid lines, separated by twice the minimum network delay, δ. At time t_0, the TICSync Estimator begins converging to slope α_1. At time t_1 the slope changes instantaneously to α_2. t_2 is the time at which a fully converged Estimator would have detected the change in slope. But in this case, until t_3, the Estimator is still able to find feasible lines of maximum separation which pass between the corridor bounds. The dashed line is the last of these. The upset is detected at t_3.

Solving for the location of t_3 yields the detection period

$$\Delta = t_3 - t_1 \tag{8.6}$$

$$= \frac{2\delta}{|\alpha 2 - \alpha 1| - \frac{2\delta}{t_1 - t_0}} \tag{8.7}$$

The detection period must be positive, which requires that

$$|\alpha 2 - \alpha 1| > \frac{2\delta}{t_1 - t_0}, \tag{8.8}$$

implying that sufficiently small slope changes will *never* be detected. The algorithm will eventually converge to the true slope, but with a constant offset error, and separation lines that are closer together than twice the minimum network delay. It is worth reiterating that the offset will still never be in error by more than 2δ.

8.2.3 Skew Change Detection - A Better Approach

In designing a filter sensitive to skew upsets, it is important to think about what constitutes an 'acceptable' delay for detection. Large changes need to be detected and adjusted to rapidly, but it is acceptable (and necessary) to take longer before trusting the detection of a small estimated change. Small errors in slope will only cause the offset error to increase slowly.

Slope changes cannot be immediately detected and a certain amount of evidence must be collected before they can be reliably determined. Suppose the skew changed instantaneously by 2 PPM; even if the filter was started afresh at the precise moment of the change, it would need to run for some time before it converged to an accuracy of better than 2 PPM. Only then would its output be useful for showing a change in skew from the previous value.

We now investigate a solution which is independent of the minimum network delay, δ. Because the TICSync Estimator updates are so cheap, it is possible to run a Bank of Estimators operating at different scales.

Suppose we decide that we wish to detect all skew changes greater than 50 PPB, and that a basic TICSync Estimator must run for W samples in order to converge to such an accuracy. An optimal filter would run a Bank of Estimators, $E_1 \,..\, E_W$ on sliding windows of every integer interval up to W. If at any point one of the Estimators detects a slope error greater than its current precision, then a skew upset is deemed to have occurred within the Estimator's window. The proposed filtering algorithm would be $O(W)$ for each update (with careful management of the Estimator Bank), resulting in $O(NW)$ overall cost for an N sample dataset. For detecting small changes with high sample rates, this could be prohibitively expensive.

A significantly cheaper solution can be achieved by running a reduced Bank of Estimators. Each Estimator uses a window twice the size of the last. Such a solution incurs a cost of only $O(\log_2 W)$ per update, with overall cost $O(N \log_2 W)$. It is

Estimator	Window length		Detectable Slope Error
	(Samples)	(Seconds)	(PPM)
E_1	20	4.0	136.39
E_2	40	8.0	51.80
E_3	80	16.0	19.67
E_4	160	32.0	7.47
E_5	320	64.0	2.84
E_6	640	128.0	1.08
E_7	1280	256.0	0.41
E_8	2560	512.0	0.15
E_9	5120	1024.0	0.06
E_{10}	10240	2048.0	0.02

Table 8.1: A typical TICSync+ (Bank-Of-Estimators) filter with 10 Estimators. Each successive Estimator runs over a longer window, but is able to detect smaller skew errors. The figures presented here are for a Weibull Distribution with $k = 2.52$ and $\lambda = 0.00035$.

sketched out in Algorithms 3, 4 and 5. In the worst case, the detection of a given skew change will take twice as long as the full Bank-of-Estimators filter. On average it will take 1.5 times as long. We call this configuration TICSync+.

In the proposed TICSync+ implementation, the TICSync Estimator E_0 is fed all measurements, and is only reset if an upset is detected. This allows it the opportunity to converge to arbitrary precision when there are no upsets. The bank of basic TICSync Estimators E_i, for $i = 1 \ldots S$ are each never allowed to exceed $w2^{i-1}$ samples in length, where w is the maximum length of the lowest order estimator, E_1. When an Estimator reaches it's maximum length, it takes a copy of the next smallest Estimator and continues adding measurements. This scheme ensures that an Estimator E_i always has a window of at least $w2^{i-2}$ samples available to it. Table 8.1 gives typical detectable skew errors for a bank of 10 Estimators.

The TICSync+ algorithm has only two user-selected parameters: The number of

TICSync Estimators, S, and the smallest window size, $w = \lceil W/2^{S-1} \rceil$. Choosing their values is a trade-off between the cost of the updates ($O(S)$), the smallest detectable skew error (where W is calculated using Equation 7.60), and the required rapidity of detection. The lack of further tuning parameters makes the algorithm very simple to deploy.

Algorithm 3 Main loop for TICSync+ Bank of Estimators Filter

```
procedure TICSync+(S, w)
    ▷ S: size of Estimator Bank
    ▷ w: window size for smallest Estimator

    for i ← 0 to S do
        E_i ← Init-Estimator()
    end for
    loop
        [d, q] ← GetNextMeasurementPair()
        Update-Estimator-Bank(E, S, w, d, q)
        Detect-Upsets(E)
    end loop
end procedure
```

Algorithm 4 Update Estimator Bank with a new measurement

```
procedure Update-Estimator-Bank(E, S, w, d, q)
    if E_1.Num-Samples() = w then
        for j ← S to 2 step −1 do
            if E_j.Num-Samples() = 2^{j−1} w then
                E_j ← E_{j−1}
            end if
        end for
        E_1.Reset()
    end if

    for i ← 0 to S do
        E_i.Update(d, q)
    end for
end procedure
```

It may seem a drastic measure to reset *all* TICSync Estimators when even a small skew change is detected. When an Estimator detects a skew change, the precise

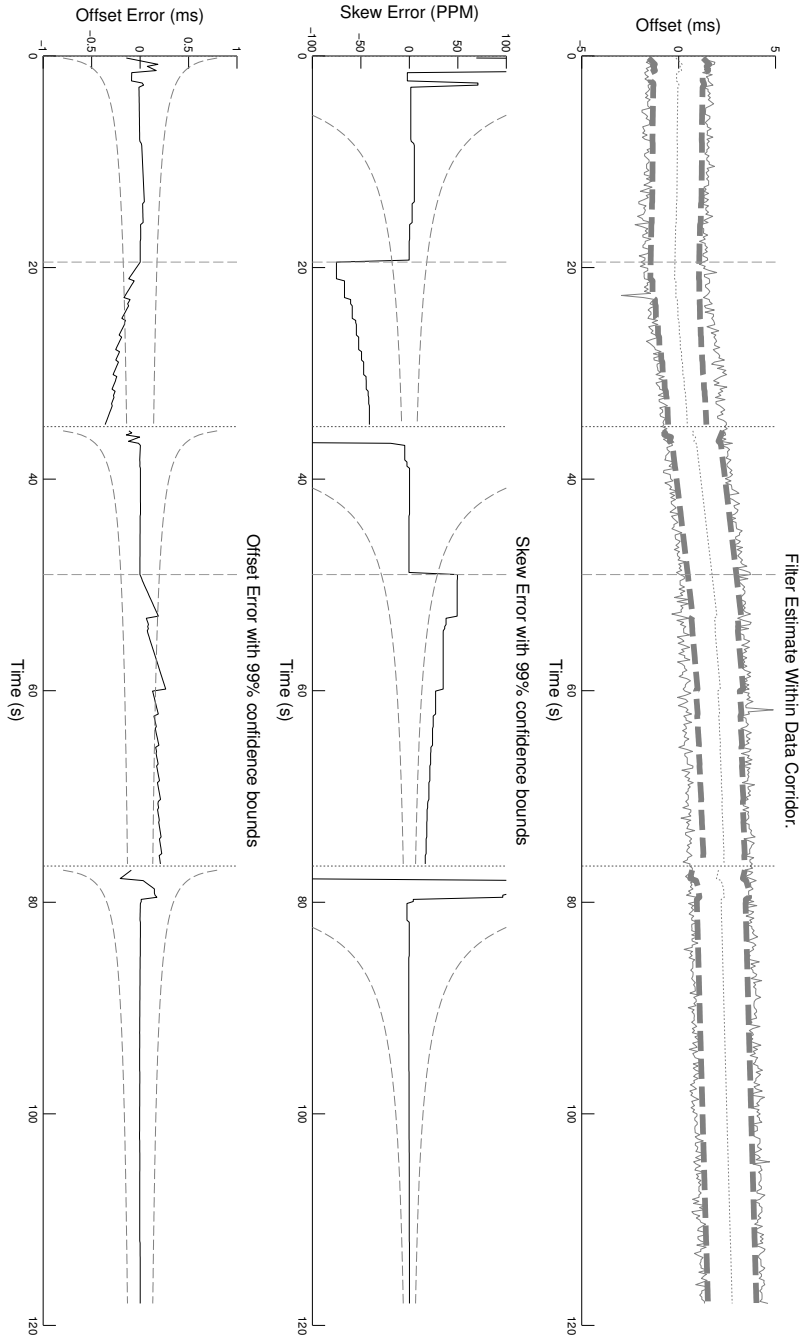


Figure 8.5: The TICSync+ Bank-Of-Estimators response to two slope change events (occurring at the vertical dashed lines). The first is a change of 75 PPM, and the second 50 PPM. The data were obtained by randomizing some real packet delay data from a local network and applying synthetic skew events. There are 4 filters in this example, running at scales of 4,8,16 and 32 seconds. The smallest theoretical error detectable by the 32 second window is 6.5 PPM in this particular case. The points at which the upsets were detected are shown by vertical dotted lines. In the periods between event and detection, the filter may be in error by more than the expected bounds. The second slope change is relatively slight, so the filter takes longer to detect it. Note that at no point does the offset error exceed 500µs.

Algorithm 5 Compare Estimators to detect Reset and Skew errors

```
procedure DETECT-UPSETS(E, S)
    if E_0.CHECKHULLINTERSECTION() then
        RESET-ESTIMATORS(E)
        return
    end if

    for i ← 1 to S do
        ε_α ← |E_i.SLOPE() − E_0.SLOPE()|
        if ε_α > E_i.MINDETECTABLESLOPEERROR() then
            RESET-ESTIMATORS(E)
            return
        end if
    end for
end procedure
```

location of the change is unknown, other than that it must have occurred somewhere in the detecting Estimator's measurement window. It could even have occurred just in the last few samples. The filter cannot be sure for any of the Estimators that the upset did not occur within the Estimator's measurement window. If E_k was the detecting Estimator, then it might seem sensible to copy the contents of E_{k-1} into E_0, and reset all Estimators larger than E_{k-1}. The idea being that E_0 would then be more correct than before, and that the remaining error would be picked up by a subsequent filter. This would hopefully result in a smoother transition. The problem with this approach is that (depending on where the upset occurred) the remaining error might be less than the filter's minimum discernable error. In that case, the measurement window of E_0 will remain spanning the upset point, and it will remain in error indefinitely.

In some cases, rapid re-initialization may be achieved by using the previous slope estimate in preference to the normal filter output until the Estimator has converged enough that its estimated error is less than the difference between the current and previous slope estimates. Alternatively, given a prediction of the number of samples required for the filter to converge to the required accuracy, the adjusted estimate

could be smoothly phased out in favour of the normal filter output. This technique works very well in some circumstances, but in others the results are much worse than just allowing the filter to start from scratch. Smoothing the filter response after an upset detection is an area requiring further research, but in the mean-time, resetting the whole filter is a conservative approach to the problem.

8.2.4 Small Reset Detection

Small resets less than 2δ in magnitude will not cause an immediate hull intersection. They may cause one of the Bank-Of-Estimators to detect a skew change, but this is not guaranteed. Once the small reset lies outside of an Estimator's sliding window, the skew will return to the previous value, making the upset undetectable. The long-term effect of the upset will be to reduce the distance between the lines of maximum separation.

Using the approximation for an upper bound on the offset error allows us to add another test to the function DETECT-UPSETS in Algorithm 5. The code snippet is shown in algorithm 6.

Algorithm 6 Offset detection snippet

```
ε_τ ← |E_i.OFFSET() − E_0.OFFSET()|
if ε_τ > E_i.MINDETECTABLEOFFSETERROR() then
    RESET-ESTIMATORS(E)
    return
end if
```

With the new test in place, the TICSync+ filter is equipped to detect both small and large resets as well as changes in skew, without being affected by network load fluctuations (which tend to vary the width of the bounds corridor). Figure 8.6 shows the typical response time for small resets, as well as a situation where a change in network load has no effect on the filter.

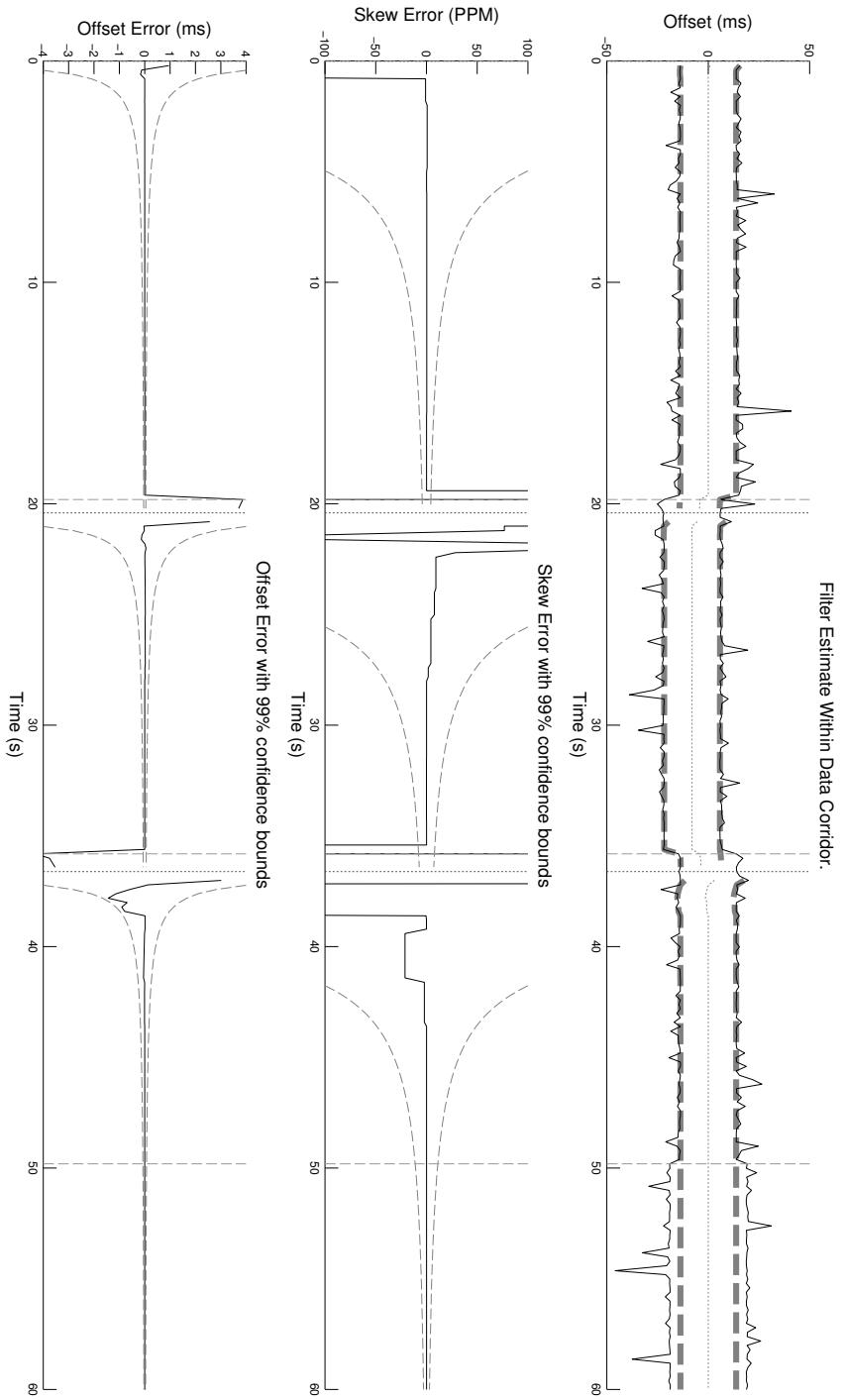


Figure 8.6: The TICSync+ Bank-Of-Estimators response to small magnitude reset events (occurring at the first two vertical dashed lines). Both have a magnitude of 8ms. The data were obtained by randomizing some real packet delay data from a local network and applying synthetic skew events. The points at which the upsets were detected are shown by vertical dotted lines. Note that smaller changes are rapidly detectable, but were not easy to see in the figure. There are 4 filters in this example, running at scales of 4,8,16 and 32 seconds. The third vertical dashed line marks a change in network load. There is no offset or skew change, so the filter ignores it.

8.2.5 Drift Detection

The linear clock model which is assumed by the TICSync Estimator does not allow for the possibility of non-zero drift. Indeed, to do so would require an entirely different approach, as convex hulls would no longer be applicable. Nonetheless, the TICSync+ Bank-Of-Estimators approach handles such events gracefully by approximating them as piecewise linear. This does affect the minimum achievable error, which will be a function of the drift magnitude. Figure 8.7 shows a typical response to an extreme constant drift situation. In the steady state the filter reset points are regularly spaced, and always prompted by the same Estimator in the Bank of Estimators.

A filter operating in the constant drift regime would certainly benefit from further research into rapid reinitialization. It might be possible to detect regular slope error events and learn the average change in slope over time. This could be fed into the Estimators to aid in initialization.

8.3 Summary

Using the insights gained in Chapter 7, we developed TICSync: a new constant time incremental algorithm for rapidly synchronizing clocks across a network to arbitrary skew accuracy. The algorithm uses a pair of convex hulls to extract salient points from incoming timing data and thus has modest memory requirements.

By modelling expected packet delay measurements with a Weibull distribution, we were able to derive analytic expressions for the probable error of the TICSync filter at any time. This led us to the idea of using a bank of TICSync Estimators running at different scales in order to detect clock upsets of various types and magnitudes as rapidly as possible. By choosing the scales carefully, we obtained good detection performance with only a limited number of Estimators. When Estimators reach their maximum window size, the data gathered is passed on to Estimators running at larger

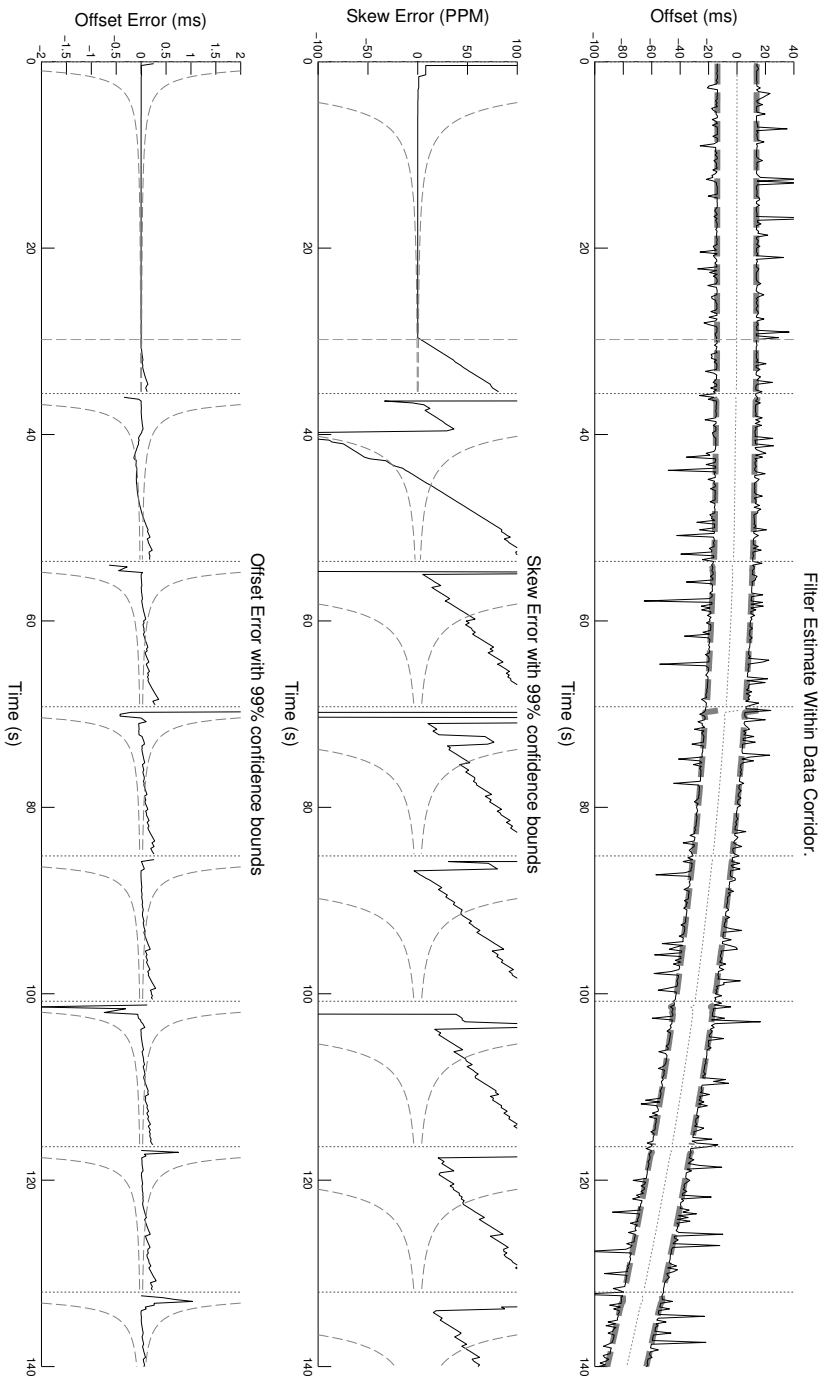


Figure 8.7: The TICSync+ Bank-Of-Estimators response to a constant-drift upset. The period of constant drift begins at the vertical dashed line, and has a magnitude of $15\mu s/s^2$. The data were obtained by randomizing some real packet delay data from a local network and applying synthetic skew events. There are 4 filters in this example, running at scales of 4,8,16 and 32 seconds. The constant drift has the effect making the slope error increase at a constant rate. The vertical dotted lines show the filter reset points. Despite the large slope errors and regular resets, the filter generally remains accurate to better than $200\mu s$. The situation depicted is much more extreme than would be expected for typical clocks. In more 'normal' situations the typical offset error will be much lower than $200\mu s$.

scales to maintain efficiency. The final algorithm, known as TICSync+ retains the constant time incremental performance of TICSync, making it suitable for long-term online deployment.

Although TICSync+ is a standalone contribution and has broad applicability, in the context of the mobile robotics endeavour, TICSync+ fulfills our requirement for a system which enables accurate data-timestamping across multiple machines. What remains to be done in the last chapter is to summarise this thesis, its structure and contributions and emphasise the underlying theme which binds it together.

Chapter 9

Summary

In this Section we will review the main contributions made by this thesis. We begin by explaining the structure, chronology and interconnectedness of its components.

9.1 Perspective

This is a thesis of two distinct parts. The first (Chapters 2 - 5) is sensor based. It is concerned with the fusion of laser and camera data and the correction, subject to an architectural prior, of 3D point clouds. The second part (Chapters 6 - 8), is about securing precision timing across a distributed network. This is motivated by the need to ensure the veracity of time sequential data for use in mobile robotics. That is not to say that these two parts are unrelated. On the contrary, the entire thesis is driven by a desire for 'better sensing', be that through the fusion of two different modalities or synchronization of data streams. The ordering of the contents of this thesis represents the flow of the research as it happened. The initial intention was to concentrate solely on generating high fidelity surfaces from point clouds. It quickly became clear that firstly the problem was totally driven by point cloud quality, and secondly that much could be gained by leveraging appearance information. The former consideration resulted in a large but worthy systems engineering endeavour involving both hardware

and software aspects. It was during this research period that the importance of good timing became apparent. Initially this issue was addressed with bespoke hardware (described in Chapter 3) and this approach allowed the development of the MRF framework which is discussed in Chapter 5 and the point cloud adjustment technique described in Chapter 4. However during the writing of papers [76] and [104] it became clear that timing errors of a just a few milliseconds between cameras and lasers were having a profound effect on data quality - a 5ms error while sensing objects 40m away during a 45°/s turn gives over 15cm of error. The investigation into the cause of such timing errors and previous work that had been done in clock synchronisation (in a community some way from mobile robotics) led the research in an unexpected direction - certainly for a mobile robotics thesis. The outcome of this introspection was a new timing algorithm (Chapters 6-8) that was shown to have advantages over commonly used approaches such as NTP, and with provable convergence and performance properties. Although the latter research was driven by a desire to achieve better sensor data time stamps, it resulted in an algorithm and a package of research which in many ways stand alone with applications far beyond laser time stamping (as it was used in [76]). For this reason we did not strive to tightly include the timing work in the sensor exploitation chapters, preferring to make explicit its independence while acknowledging its provenance.

9.2 Contributions

In this thesis, we have explored many aspects of the data acquisition and processing pipeline on a mobile robot. By analyzing a typical surface reconstruction scheme in Chapter 2, we discovered the importance of accurate and consistently sampled point clouds for high quality reconstructions. Noisy measurements can lead to poor surface normal estimates, which are of vital importance in surface reconstruction. At every

stage of the pipeline, we have analyzed the causes of errors and applied a careful treatment to minimize their impact on the data. We now précis the contributions made in the preceding chapters.

Laser Hardware and Calibration Methods We described a low cost solution for gathering 3D point cloud data from a mobile robot, using a commercial 2D laser scanner mounted on a reciprocating mechanical nodder unit. The unit allows for rapid data collection and produces good coverage of the environment even with the robot travelling at reasonable speed. Having no direct control over the nodding unit means that we cannot sidestep the issue of time stamping, which is a particular problem with high-speed data acquisition. To deal with the problem, we developed a simple calibration procedure that runs at the beginning of each data gathering experiment, transparently to the user. It learns the timing offset between laser data and nodder encoder measurements by means of an optimization that minimizes the discrepancy between data gathered on upward and downward sweeps of the nodder.

We have found the nodding laser to produce very high quality point clouds which are an ideal input to the data processing pipeline.

Data Processing Requiring the robot to be stationary during the gathering of high quality point clouds places unacceptable constraints on the data collection process. Much more attractive is the ability for the robot to be able to gather good data whilst on the move. Previous methods have restricted the robot to a reduced set of manoeuvres which minimize odometry errors, but we have tackled the problem of dealing with arbitrary vehicle motion and the associated odometry errors.

The use of architectural priors enables us to recover the vehicle trajectory by registering sparse subsets of the gathered data. This has the benefit of providing extra roll and pitch information which was previously unavailable from odometry, allowing us to deal with uneven terrain.

The method undoubtedly improves the maps that we can produce, but sometimes does not behave as expected. In areas where the architectural priors do not hold, there can be insufficient information for the algorithm to recover the vehicle's trajectory. A more general method for matching sparsely sampled point clouds in 6-DOF would make the method more widely applicable. The work appeared in the ICRA'08 conference and was a finalist for the best student paper award [46].

MRF Methods for Laser and Vision Fusion The density of point clouds produced by our nodding laser system is directly related to the amount of time available for scanning the environment. Rather than impose data gathering constraints on the robot, we looked at methods of inferring extra range data by fusing the sparse range measurements with higher resolution camera images to produce dense range images. Our main contribution was to introduce the use of 2nd-Order smoothness priors into an existing MRF framework for range interpolation. Our new prior makes effective use of image colour information to choose the most appropriate form of range inference from neighbouring nodes - either interpolation or if required, extrapolation.

We also provided a novel way to overcome errors introduced by the non-linearities of the projective camera model. The problem is not treated by existing literature, but we show how the MRF weights can be modified so that the optimization takes the camera model into account. We also give a method of down-weighting the influence of noisy range measurements that often occur at depth discontinuities.

We have shown that when properly tuned, our method is able to consistently outperform a number of recently proposed techniques running on real data. We made use of readily available datasets to benchmark the performance.

There are however a number of areas where further work could be done. In particular, a method of adaptively tuning the MRF's parameters in different areas of the image would improve results further. Currently the parameters must be set

to give reasonable performance in a wide variety of situations. A means of applying machine learning techniques to learn what parameters work well in different regimes would be highly beneficial.

The use of robust kernels in the MRF's data cost term would alleviate the problem of noisy data causing disruptions in the inferred range surface. Currently a laser measurement far from the surface carries a disproportionately high cost, due to the squared distance metric. A Huber kernel [54] or similar might produce a more appropriate cost function, though it would preclude the use of the fast linear solving techniques that we currently employ.

In order to make good use of the surfaces produced by the MRF method, we need a way of fusing or stitching surfaces from different images. There are many existing methods for fusing range images, but we are not currently able to enforce continuity between surfaces in overlapping areas. It may simply be a case of taking a surface lying between the two estimates, but further investigation is necessary.

Clock Synchronization Synchronizing the clocks of network distributed computers is vital for many tasks. In the robotics context, the problem is often dealt with in an unsatisfactory way, by running synchronization tools designed for long term stability rather than rapid convergence. We have presented a new algorithm, TICSync+, which is capable of synchronizing clocks to millisecond accuracy after just a few seconds. Given a few minutes, it can achieve synchronization performance measured in Parts Per Billion on standard PC hardware. The upset detection capability of TICSync+ allows it to be applied to computers whose clocks are being regularly interfered with by other synchronization schemes. Even in these cases it is able to robustly maintain a mapping between clocks. The algorithm requires no special hardware or modified device drivers, as it works over standard network protocols.

TICSync+ is very efficient, with constant time incremental updates and a modest

memory footprint. This makes it applicable in many areas, including embedded systems and low-power wireless sensor networks.

9.3 Future Work

We have highlighted throughout the thesis various points of extension for future work but there are some stand out cases which deserve particular attention and summary.

Laser Sensor Modelling What has not been addressed in this work is an empirical development of a full and realistic sensor model for the laser. There is likely to be benefit in future work that considers how range measurements are affected not just by the geometry of the scene (angle of incidence), but the material qualities. Indeed, one might consider a sophisticated model that leverages appearance information gleaned from images to enable the construction of a time-varying sensor model. Regretfully, due to time constraints, this topic is left for future derivative work.

Large Scale MRF Application As it stands, we used off-the shelf optimisation tools for the linear solve required in the MRF work. Further work could benefit from explicit exploitation of the structure of the problem when trying to solve over very large point clouds. The problem of fusing adjacent inferred range images would also need to be addressed before large scale dense maps could be built with the method.

Explicit Handling of Range Data Outliers We did not consider in any great detail how to handle the presence of outliers in range data. M-estimators are commonly used in vision based optimisation problems to seamlessly handle rogue data and given more time we would have taken that approach here. This would be the first extension we would undertake in future work.

TICSync There are areas in which the TICSync algorithm would benefit from further work. Currently, when an upset is detected, a conservative approach is taken and the TICSync estimators are all reset. Resynchronization is rapid, but it may take some time to achieve the levels of accuracy that had been reached before the reset. It should be possible to make use of information gathered prior to the upset detection event in order to speed up the resynchronization process. Further work will be put in to online learning of appropriate delay distribution models. The current heuristic approach usually gives a conservative estimate of error, but no guarantees can be made. It is also necessary to further consider the appropriate course of action to take when the network delay distribution changes, potentially invalidating future error estimates.

An earlier version of TICSync has been successfully running as part of the MOOS middleware system for almost 2 years, providing a time synchronization service for MOOS clients. Future work will seek to apply the algorithm more widely to our robotic platforms, with a particular effort to quantify achievable improvements in laser data from push-broom laser configurations (as used by the 'Lisa' robot introduced in Chapter 1) and rapidly rotating laser configurations. Finally, we are exploring a number of interesting applications of skew estimation algorithms in the temporal fusion of data from disparate sensors.

Appendix A

Least squares plane fitting

Let $P = \mathbf{p}_1, ..., \mathbf{p}_m$ be a set of points in $\mathbb{R}^3$. We wish to fit a plane such that the sum of squared distances from the plane to each of the points is minimized.

The equation of a plane is

$$ax + by + cz + d = 0 \tag{A.1}$$

where a, b, c, d are the free variables. The distance of a point (x, y, z) from the plane is then defined as

$$\text{dist} = \frac{ax + by + cz + d}{\sqrt{a^2 + b^2 + c^2}} \tag{A.2}$$

Our aim is to choose a, b, c, d in order to minimize the sum of squared distances to each point $\mathbf{p}_i(x_i, y_i, z_i)$:

$$f(a, b, c, d) = \frac{\sum_{i=1}^{m}(ax_i + by_i + cz_i + d)^2}{a^2 + b^2 + c^2} \tag{A.3}$$

Finding the partial differential of A.3 wrt d gives

$$\frac{\partial f}{\partial d} = \frac{1}{a^2 + b^2 + c^2} \sum_{i=1}^{m} 2(ax_i + by_i + cz_i + d) \tag{A.4}$$

Setting it to zero shows us that the minimum occurs when

$$d = -\frac{1}{m}\sum_{i=1}^{m}(ax_i + by_i + cz_i) \tag{A.5}$$

$$d = -(a\bar{x} + b\bar{y} + c\bar{z}), \tag{A.6}$$

where $\bar{\mathbf{p}}(\bar{x}, \bar{y}, \bar{z})$ is the centroid of the points in P.

If we define $\tilde{\mathbf{p}}_i(\tilde{x}_i, \tilde{y}_i, \tilde{z}_i) = \mathbf{p}_i - \bar{\mathbf{p}}_i$, then A.3 may be rewritten as

$$f(a,b,c) = \frac{\sum_{i=1}^{m}(a\tilde{x}_i + b\tilde{y}_i + c\tilde{z}_i)^2}{a^2+b^2+c^2} \tag{A.7}$$

$$= \frac{1}{a^2+b^2+c^2}\sum_{i=1}^{m}(a^2\tilde{x}_i^2 + b^2\tilde{y}_i^2 + c^2\tilde{z}_i^2 + 2ab\tilde{x}_i\tilde{y}_i + 2ac\tilde{x}_i\tilde{z}_i + 2bc\tilde{y}_i\tilde{z}_i) \tag{A.8}$$

$$= \frac{1}{a^2+b^2+c^2}\begin{bmatrix} a & b & c \end{bmatrix}(\mathbf{M}^T\mathbf{M})\begin{bmatrix} a \\ b \\ c \end{bmatrix} \tag{A.9}$$

where M is defined as

$$M = \begin{bmatrix} \tilde{x}_1 & \tilde{y}_1 & \tilde{z}_1 \\ \tilde{x}_2 & \tilde{y}_2 & \tilde{z}_2 \\ \vdots & \vdots & \vdots \\ \tilde{x}_n & \tilde{y}_n & \tilde{z}_n \end{bmatrix} \tag{A.10}$$

If we further define

$$v = \begin{bmatrix} a & b & c \end{bmatrix}^T \tag{A.11}$$

then A.9 becomes the Rayleigh quotient,

$$f(v) = \frac{v^T(M^T M)v}{v^T v} \tag{A.12}$$

The Rayleigh quotient is minimized by setting v to be the eigenvector of $M^T M$ with the smallest eigenvalue. In practice the Singular Value Decomposition of M gives the eigenvectors of $M^T M$ and the squares of the eigenvalues of $M^T M$, but is more efficient to compute than the eigen decomposition of $M^T M$.

To recap, the normal vector of the best-fit plane is given by the eigenvector $\mathbf{u}_0$ having smallest eigenvalue λ_0. The two other eigenvectors lie within the plane. If the plane fit is perfect then the magnitude of the smallest eigenvalue will be 0.

Appendix B

KD-Trees

Given an unorganized point cloud $X = \{\mathbf{x}_1, ...\mathbf{x}_n\}$, the most obvious data structure to represent them would be a point array of length n, but it does not explicitly convey any information about the structure of the point cloud. That is, there is no information stored about the relationships between the points

Many surface reconstruction algorithms need to make repeated 'neighbourhood' queries of the point cloud. Given an arbitrary query location $\mathbf{q}$, the goal of a neighbourhood query is to find the k points in X that are closest to $\mathbf{q}$. When the data structure is a simple array, such a query requires calculation of the Euclidean distance from $\mathbf{q}$ to every point in X, followed by a sort on distance. This is an $O(n + k \log k)$ operation, but by using an efficient data structure the complexity can be reduced to $O(k \log n)$.

By using a data structure optimized for neighbourhood searches, surface reconstruction algorithms can be significantly more efficient. One such data structure is the kd-tree, introduced in [40]. The kd-tree is a binary tree structure that allows neighbourhood searches to be performed in logarithmic time and can be used in conjunction with an array-type representation. It should be noted that kd-trees can be applied to data of arbitrary dimension, though they become less efficient as the num-

ber of dimensions increases. We are concerned only with data in $\mathbb{R}^3$, where efficiency is high.

Building the Tree

Building a kd-tree requires recursive division of the space occupied by the point cloud into a series of hyper-rectangles known as cells. Starting with a cell containing all points in X, a division is made perpendicular to one of the principal axes. The location of the division depends on the exact splitting rule being used, but a common rule is to divide the cell along its longest dimension, at the mean of the point coordinates in that direction. A node is added to the kd-tree, containing information about the split direction and the dimensions of the two new cells.

Following the cell split, the division function is recursively called for each of the two new cells. Divisions performed in those cells are added to the kd-tree as sub-nodes of the parent division node.

The recursion terminates when a cell contains only a single data point. A *leaf* node is added to the kd-tree under the parent division node, containing a reference to that data point.

The tree building step is $O(n \log n)$, requiring $O(n)$ storage. A disadvantage to the kd-tree's structure is that it is non-trivial to add new points once it is built. Of course, it is possible to find the cell in which the new point lies and perform another division step, but it can result in a poorly balanced tree if many points are added in the same area. In these cases the structure becomes increasingly inefficient and it is usually best to simply rebuild the whole tree.

Querying the Tree

To find the k nearest neighbours of an arbitrary query point, $\mathbf{q}$, the algorithm proceeds as follows:

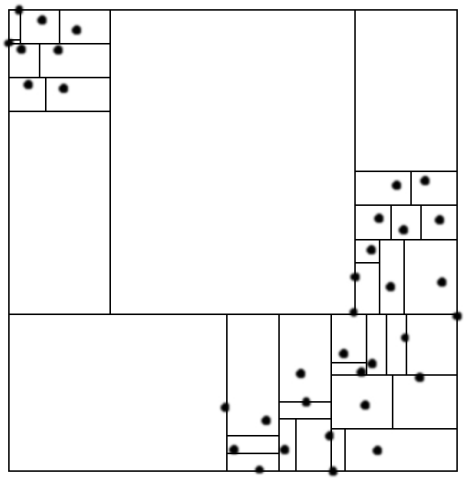


Figure B.1: A kd-tree partitioning of a synthetic 2D point cloud. Each partition line corresponds to a branch node and each cell corresponds to a leaf node in the kd-tree data structure.

1. Initialize a neighbourhood set $N_q = \emptyset$, to be sorted by distance from $\mathbf{q}$

2. Traverse tree to find the cell C_q in which $\mathbf{q}$ lies.

3. Traverse tree in the neighbourhood of the node C_q until k points have been encountered.

   Populate N_q with the k best candidates seen so far, sorted in ascending distance from $\mathbf{q}$.

4. Inspect all unvisited cells that intersect a hypersphere centred on q, with radius just touching $\mathbf{x'}_k$. If points are encountered closer than those in N_q, then insert them into N_q, replacing the most distant points and then re-sort.

For details of different cell splitting schemes, and a description of *approximate* kd-trees for faster neighbourhood queries, see [74].

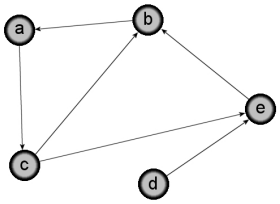
Appendix C

Graphs and Spanning Trees

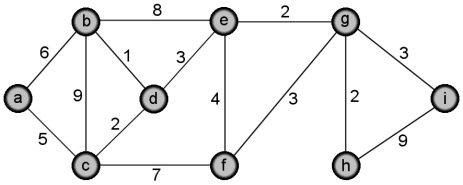
A graph $G = (V, E)$ is a data structure consisting of a set of *vertices* (or *nodes*), V and a set of *edges*, E. An edge in E is defined as a pair (u, v) where $u, v \in V$ and $u \neq v$. In an *undirected* graph the vertex pairs are unordered, whilst in a *directed* graph the pair is ordered, and signifies that the edge may only be traversed in one direction. Examples of both types can be found in Figure C.1.

There are many data structures which are more specialized forms of graphs, including trees, linked lists, heaps and priority queues. In particular, a *tree* is defined as a connected, acyclic, undirected graph. The term 'connected' means that there is some path by which any vertex may be reached from any other vertex. 'acyclic' means that the graph contains no loops.

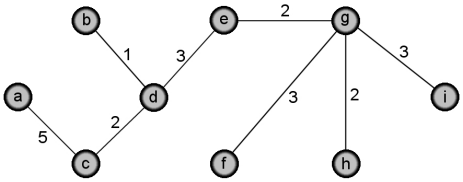
In many graph implementations, it is convenient to attach a *weight* or *cost* to each edge in E. This may represent a physical distance, or perhaps a penalty associated with the traversal of that edge. Then the *total cost* of the graph is the sum of all the edge costs.



(a) A directed graph



(b) An undirected graph, with edge costs



(c) A Minimum-Spanning-Tree (MST) of C.1b

Figure C.1: Some simple graphs

C.1 Minimum Spanning Trees

If G is a connected, undirected graph then a *spanning tree* of G is an undirected, acyclic, connected graph which contains all the vertices of G but a subset of the edges. G may have multiple spanning trees.

A *minimum spanning tree* (MST) of G is a spanning tree of minimum total cost. That is, a tree whose edges are a subset of E and which connect all the vertices of G, but that has a total cost at least as small as any other spanning tree of G. There may be multiple MSTs that share the same minimum cost but use different edges.

There are a number of algorithms for finding MSTs (see [26]). The algorithm of Prim is described here, and it proceeds as follows:

- Initialize a tree, T with an arbitrarily chosen vertex from the graph as its only node
- Place all edges in G into a set, E
- Loop until E is empty (or T spans all vertices)
  - Find the edge, u having minimum weight in E

  - If u connects a vertex in T with a vertex not in T
    - Add u and the new vertex to the tree, T
  - Remove u from E
- T is now a minimum spanning tree of G

Appendix D

Marching Cubes

Marching Cubes [62] is an algorithm for extracting *isosurfaces* from scalar fields. The output is a triangular mesh which approximates the chosen isosurface. To produce the mesh, the scalar field is divided into a regular grid of cubes, and each cube is then replaced in turn with one of 256 preset mesh components which best matches the isosurface as it passes through that cube. The name *marching cubes* comes from the way in which the algorithm *marches* through the entire grid of cubes.

To illustrate the method, consider the more simple case of marching squares. First, the area is divided into a grid of equal squares. The scalar field is evaluated at the four corners of each square to determine whether its value is *above* or *below* the value of the isocontour to be extracted. In Figure D.2a, the red blob represents areas of the scalar field which are above the value, so that the isosurface to be extracted lies at the boundary of the blob. The grid nodes (where the function is evaluated by the Marching Cubes algorithm) are coloured black or white, to denote *above* or *below* respectively.

Since each square has four nodes, there are only 16 possible configurations. The algorithm simply looks up the case in a list (Figure D.1) and approximates the isocontour with a small straight line segment. By repeating this for every square, a

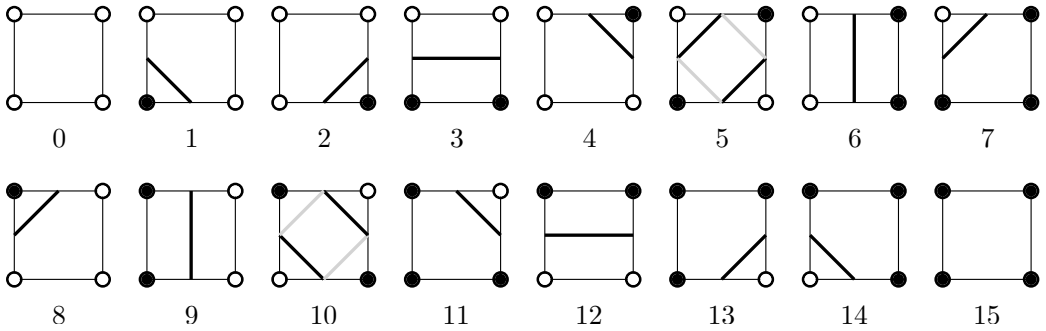
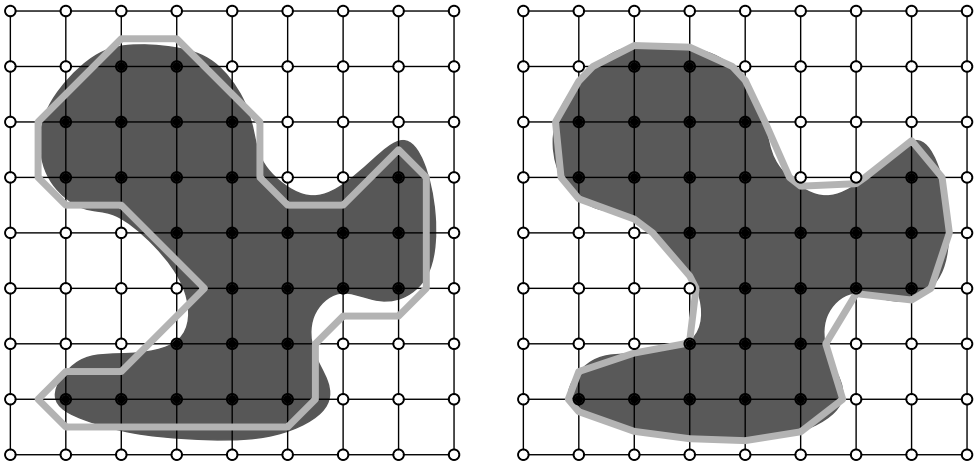


Figure D.1: The 16 Marching Squares cases. The grey lines in cases 5 and 10 denote ambiguous configurations, though either one will create a valid closed surface.



(a) Marching Squares **(b)** Adaptive marching squares

Figure D.2: The required contour is the boundary of the dark blob. The thick line is the approximation created. Notice how the adaptive method pushes the vertices closer to the true contour.

piecewise linear representation of the isocontour is constructed. This is represented by the green line in Figure D.2a.

The algorithm is $O(mn)$, where m, n are the grid dimensions. The quality of fit can be improved by using a finer grid, but this brings with it higher computational cost, increased memory requirements and more evaluations of the scalar field. It may be noted that many of the vertices of the contour approximation are a long way from the true contour. A better fit can be achieved by allowing the vertices to slide along the grid line section that they are attached to. Since the value of the scalar field is known at both ends of the grid line segment, a simple linear interpolation can be

performed to determine an intersection point closer to the true contour value. Figure D.2b shows the results of the technique, which is known as *adaptive marching squares*.

The Marching Cubes algorithm is simply an extension of the Marching Squares algorithm into 3D. With 8 nodes per cube, there are 256 cases. There are 15 major cases, with all the remaining cases being rotations of the major cases.

Notice that cases 5 and 10 in Figure D.1 are ambiguous, with two possible contour approximations. Provided that one is consistent in the representation used, the contours created are not topologically incorrect. There are similarly ambiguous cases for marching cubes but they may be topologically incorrect. A strategy for detecting and fixing the ambiguities is described in [80].

Further speed improvements can be achieved by avoiding evaluation of grid cubes which do not contain the surface. By seeding the algorithm somewhere on the surface, it can then *follow* the surface to be extracted by only evaluating cubes through which the surface subsequently passes. This can be problematic if the surface is not fully connected.

Many subsequent modifications (ie. [100, 57, 96]) have been introduced to improve Marching Cubes, but the original algorithm is currently adequate for our purposes.

Appendix E

Spherical k-means Algorithm

E.1 k-means

k-Means is a technique used to segment a set of point samples into k clusters, where each data point is assigned to its closest cluster centre. Let $X = \{x_1 \,..\, x_n\}$ be a set of point samples with $X \in \mathbb{R}^3$. The aim is to partition X into k clusters $\Theta = \{\Theta_1 \,..\, \Theta_k\}$, to minimize the expression

$$\arg\min_{\Theta} \sum_{i=1}^{k} \sum_{x_j \in \Theta_i} \|x_j - \mu_i\|^2, \qquad \text{where } \mu_i = \frac{1}{|\Theta_i|} \sum_{x_j \in \Theta_i} x_j \tag{E.1}$$

There are a few algorithms for solving the problem; the most popular, known as Lloyd's algorithm is described by Bishop [13].

E.2 Spherical k-means Algorithm

The spherical k-means algorithm due to Dhillon and Modha [31] is a minor modification of Lloyd's algorithm. It is used to find cluster centres of points lying on a spherical manifold of unit radius, using a squared cosine similarity metric rather than the squared euclidean distance. The algorithm uses the result that the central point

of a cluster in a cosine-distance sense is given by the normalized Euclidean mean of the points assigned to the cluster. The cosine similarity between two normalized vectors is easily computed with the dot product operation.

Another way of putting this is that if $\hat{c}$ is the normalized Euclidean mean of X, then there is no vector $\hat{z}$ for which the quantity $\sum_{i=1}^{n} x_i^T \hat{z}$ is greater than $\sum_{i=1}^{n} x_i^T \hat{c}$. Formally,

$$\sum_{i=1}^{n} x_i^T \hat{z} \leq \sum_{i=1}^{n} x_i^T \hat{c} \qquad \forall \quad \hat{z} = \frac{z}{\|z\|}, \quad z \in \mathbb{R}^3 \tag{E.2}$$

where

$$\hat{c} = \frac{\bar{x}}{\|\bar{x}\|} \quad \text{and} \quad \bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{E.3}$$

The inequality is proved by manipulating equation E.2 to become

$$\sum_{i=1}^{n} \frac{x_i^T z}{\|z\|} \leq \sum_{i=1}^{n} \frac{x_i^T \bar{x}}{\|\bar{x}\|} \tag{E.4}$$

$$n \frac{\bar{x}^T z}{\|z\|} \leq n \frac{\bar{x}^T \bar{x}}{\|\bar{x}\|} \tag{E.5}$$

$$\bar{x}^T z \leq \|\bar{x}\| \, \|z\| \tag{E.6}$$

which is a form of the Cauchy-Schwarz inequality. In this representation, $\bar{x}^T z$ is a dot product, so that

$$\|\bar{x}\| \, \|z\| \cos\theta \leq \|\bar{x}\| \, \|z\| \tag{E.7}$$

which must be true because $\cos\theta \leq 1$.

E.2.1 The Algorithm

The algorithm (reproduced here as Algorithm 7) takes as input the set of points $X = \{x_1 \ldots x_n\}$ and initial cluster centres, $M^0 = \{\mu_1^0 \ldots \mu_k^0\}$. It iterates until the cluster assignments are stable, and returns the final cluster centres.

Algorithm 7 Spherical k-means

procedure SPKMEANS($X = \{x_1 .. x_n\}$, $M^0 = \{\mu_1^0 .. \mu_k^0\}$)
$M \leftarrow \{\mu_1^0 .. \mu_k^0\}$
$\Theta \leftarrow \{\}$

repeat
$\Theta^{old} \leftarrow \Theta$

▷ Assign each point to closest cluster centre
for $j \leftarrow 1$ to n **do**
$A_j \leftarrow \arg\min_i \left(x_j^T \mu_i\right)$
end for

▷ Recompute cluster centres
for $i \leftarrow 1$ to k **do**

$\Theta_i \leftarrow \{x_j \mid 1 \leq j \leq n, A_j = i\}$

$\mu_i \leftarrow \dfrac{\text{mean}(\Theta_i)}{|\text{mean}(\Theta_i)|}$

end for

until $\Theta^{old} = \Theta$

return $\{\mu_1 .. \mu_k\}$

end procedure

Appendix F

The Modified Moment Estimator

The 3-parameter Weibull distribution has the pdf

$$p(t;\lambda,k,\rho) = \begin{cases} \frac{k}{\lambda}\left(\frac{t-\rho}{\lambda}\right)^{k-1} e^{-\left(\frac{t-\rho}{\lambda}\right)^{k}} & t \geq \rho \\ 0 & t < \rho \end{cases}, \tag{F.1}$$

where k is the shape parameter, λ the scale parameter and ρ the position parameter.

Cohen and Whitten [24] provide an algorithm for the estimation of the 3 parameters given a set of samples $\{x_0 \ldots x_n\}$ drawn from the distribution. The algorithm does not operate on the sample values themselves. Rather it requires nothing but three population statistics: the mean, the variance and the minimum value. This property makes it particularly suitable for use in algorithms requiring online Weibull parameter learning.

The derivation is somewhat involved and will not be reproduced here, but it results in a few simple steps to obtain the parameter estimates $\hat{k}, \hat{\lambda}$ and $\hat{\rho}$:

1. Calculate $s^2, \bar{x}$ and $x_{(1)}$ which are the sample variance, sample mean and first order statistic (minimum sample value) respectively.
2. Set $W = \dfrac{s^2}{(\bar{x} - x_{(1)})^2}$

3. Obtain the shape parameter estimate $\hat{k}$ by finding the root of the equation,

$$f(\hat{k}) = \frac{\hat{\Gamma}_2 - \hat{\Gamma}_1^2}{\left[(1 - N^{-1/\hat{k}})\hat{\Gamma}_1\right]^2} - W, \tag{F.2}$$

where $\hat{\Gamma}_i = \Gamma\left(1 + i/\hat{k}\right)$, Γ is the Gamma function, and N is the number of samples. f is a function only of $\hat{k}$.

4. Compute the scale and position parameter estimates,

$$\hat{\lambda} = \frac{s}{\sqrt{\hat{\Gamma}_2 - \hat{\Gamma}_1^2}} \tag{F.3}$$

$$\hat{\rho} = \bar{x} - \hat{\lambda}\hat{\Gamma}_1 \tag{F.4}$$

Step 3 requires an iterative root-finding algorithm; we used the Newton-Raphson method which requires the derivative of the function $f(\hat{k})$. After a great deal of headscratching and coffee, we derived this to be

$$f'(\hat{k}) = \frac{2}{\left(k\hat{\Gamma}_1 g(\hat{k})\right)^2}\left[\hat{\psi}_1\hat{\Gamma}_1^2 - \hat{\psi}_2\hat{\Gamma}_2 + (\hat{\Gamma}_2 - \hat{\Gamma}_1^2)\left(\hat{\psi}_1 + \frac{\left(1 - g(\hat{k})\right)\log(N)}{g(\hat{k})}\right)\right] \tag{F.5}$$

where $g(\hat{k}) = 1 - N^{-1/\hat{k}}$, $\hat{\psi}_i = \psi\left(1 + i/\hat{k}\right)$ and ψ is the Digamma function, defined as $\psi(x) = \dfrac{\Gamma'(x)}{\Gamma(x)}$. Efficient solvers for the Gamma and Digamma functions are provided by Cody [22] and Bernardo [10].

Appendix G

A Running Variance Estimator

The formula for computing the unbiased variance of a sample population of size n is well-known:

$$\sigma^2 = \frac{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2}{n(n-1)} \tag{G.1}$$

However, computing it naïvely can lead to significant numerical errors. Floating point representations of numbers have a limited number of significant figures available to them. The sum-of-squares and square-of-sums quantities can become very large, though their difference may be small. There may be insufficient significant figures available to obtain the difference between the numbers to the required precision. In some cases this can lead to a negative value as the result of the computation.

In his second book in the *Art of Computer Programming* series [56], Knuth briefly mentions an algorithm for computing the variance of a set of samples in an incremental fashion, that is much more robust to numerical precision issues. It computes the variance in a single pass, making it useful for maintaining incremental running mean and variance measures. Algorithm 8 shows the method. M is used as an accumulator variable which can be divided by $n-1$ at any time to obtain the current variance.

Algorithm 8 Single-pass variance algorithm

procedure COMPUTE-VARIANCE($X = \{x_1 \,..\, x_n\}$)
    $n \leftarrow 0$ ▷ Sample counter
    $\mu \leftarrow 0$ ▷ Running mean
    $M \leftarrow 0$
    for each $x \in X$ **do**
        $n \leftarrow n + 1$
        $\delta \leftarrow x - \mu$
        $\mu \leftarrow \mu + \delta/n$
        $M \leftarrow M + \delta(x - \mu)$
    end for
    return $M/(n-1)$
end procedure

Bibliography

[1] Amenta, N., Choi, S., Dey, T. K., and Leekha, N. A simple algorithm for homeomorphic surface reconstruction. *International Journal of Computational Geometry and Applications 12*, 1-2 (2002), 125–141.

[2] Amenta, N., Choi, S., and Kolluri, R. K. The power crust, unions of balls, and the medial axis transform. *Computational Geometry 19*, 2-3 (2001), 127–153.

[3] Andreasson, H., Triebel, R., and Burgard, W. Improving plane extraction from 3D data by fusing laser data and vision. In *Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2005)* (Alberta, Canada, 2005), IEEE.

[4] Andreasson, H., Triebel, R., and Lilienthal, A. Vision-based interpolation of 3D laser scans. In *Proc. International Conference on Autonomous Robots and Agents (ICARA)* (2006).

[5] Andreasson, H., Triebel, R., and Lilienthal, A. J. *Non-iterative Vision-based Interpolation of 3D Laser Scans*, vol. 76 of *Studies in Computational Intelligence*. Springer, Germany, August 14 2007, ch. Autonomous Robots and Agents, pp. 83–90.

[6] Aweya, J., Montuno, D. Y., Ouellette, M., and Felske, K. Clock recovery based on packet inter-arrival time averaging. *Computer Communications 29*, 10 (2006), 1696–1709.

[7] Aweya, J., Montuno, D. Y., Ouellette, M., and Felske, K. Clock synchronization using a linear process model. *Int. Journal of Network Management 16*, 1 (2006), 3–28.

[8] Barber, C. B., Dobkin, D. P., and Huhdanpaa, H. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software 22*, 4 (1996), 469–483.

[9] Bauer, J., Karner, K., Schindler, K., Klaus, A., and Zach, C. Segmentation of building models from dense 3D point-clouds. In *27th Workshop of the Austrian Association for Pattern Recognition 2003, Laxenburg* (2003).

[10] Bernardo, J. M. Algorithm AS 103: Psi (digamma) function. *Journal of the Royal Statistical Society. Series C (Applied Statistics) 25*, 3 (1976), 315–317.

[11] Besl, P. J., and McKay, N. D. A method for registration of 3-D shapes. *IEEE Trans. Pattern Anal. Machine Intell. 14* (1992), 239–256.

[12] Bi, J., Wu, Q., and Li, Z. On estimating clock skew for one-way measurements. *Computer Communications 29*, 8 (2006), 1213–1225.

[13] Bishop, C. M. *Pattern Recognition and Machine Learning (Information Science and Statistics)*, 1st ed. Springer, October 2007.

[14] Bloomenthal, J. An implicit surface polygonizer. In *Graphics Gems IV* (1994), Academic Press, pp. 324–349.

[15] Bosse, M., and Zlot, R. Continuous 3D scan-matching with a 2d spinning laser. In *IEEE Int. Conf. on Robotics and Automation* (Kobe, Japan, May 2009).

[16] Bouguet, J. Y. Camera calibration toolbox for matlab, 2008.

[17] Carr, J. C., Beatson, R. K., Cherrie, J. B., Mitchell, T. J., Fright, W. R., McCallum, B. C., and Evans, T. R. Reconstruction and representation of 3D objects with radial basis functions. In *SIGGRAPH 2001, Computer Graphics Proceedings* (2001), E. Fiume, Ed., ACM Press / ACM SIGGRAPH, pp. 67–76.

[18] Carr, J. C., Beatson, R. K., McCallum, B. C., Fright, W. R., McLennan, T. J., and Mitchell, T. J. Smooth surface reconstruction from noisy range data. In *GRAPHITE '03: Proceedings of the 1st international conference on Computer graphics and interactive techniques in Australasia and South East Asia* (New York, NY, USA, 2003), ACM Press, pp. 119–ff.

[19] Chaudhari, Q. M., and Serpedin, E. Clock estimation for long-term synchronization in wireless sensor networks with exponential delays. *EURASIP J. Adv. Signal Process 2008* (2008), 27.

[20] Chaudhari, Q. M., Serpedin, E., and Qaraqe, K. A. On maximum likelihood estimation of clock offset and skew in networks with exponential delays. *IEEE Transactions on Signal Processing 56*, 4 (2008), 1685–1697.

[21] Cobzas, D., Zhang, H., and Jägersand, M. A comparative analysis of geometric and image-based volumetric and intensity data registration algorithms. In *ICRA* (2002), pp. 2506–2511.

[22] Cody, W. J. An overview of software development for special functions. In *Proceedings of the Dundee Conference on Numerical Analysis* (1975), vol. 506/1976, Springer Berlin / Heidelberg, pp. 38–48.

[23] Cohen, A. C., and Whitten, B. *Parameter estimation in reliability and life span models*, 1st ed. STATISTICS: textbooks and monographs. Marcel Dekker, Inc, New York, September 1988.

[24] Cohen, C. A., and Whitten, B. Modified maximum likelihood and modified moment estimators for the three-paramter weibull distribution. *Communications in Statistics - Theory and Methods 11*, 23 (1982), 2631–2656.

[25] Cole, D. M., and Newman, P. M. Using laser range data for 3D SLAM in outdoor environments. In *IEEE Int. Conf. on Robotics and Automation (ICRA'06)* (Orlando USA, May 2006).

[26] Cormen, T. H., Stein, C., Rivest, R. L., and Leiserson, C. E. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2001.

[27] Criminisi, A., Reid, I. D., and Zisserman, A. Single view metrology. *International Journal of Computer Vision 40*, 2 (2000), 123–148.

[28] Cristian, F. Probabilistic clock synchronization. *Distributed Computing 3*, 3 (1989), 146–158.

[29] Curless, B., and Levoy, M. A volumetric method for building complex models from range images. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1996), ACM Press, pp. 303–312.

[30] Dempster, A., Laird, N., and Rubin, D. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B 39* (1977), 1–38.

[31] Dhillon, I. S., and Modha, D. S. Concept decompositions for large sparse text data using clustering. *Machine Learning 42*, 1 (2001), 143–175.

[32] Diebel, J., and Thrun, S. An application of markov random fields to range sensing. In *Proceedings of Conference on Neural Information Processing Systems (NIPS)* (Cambridge, MA, 2005), MIT Press.

[33] Dodson, B. *The Weibull Analysis Handbook*. ASQ Quality Press, Milwaukee, April 2006.

[34] Duda, A., Harrus, G., Haddad, Y., and Bernard, G. Estimating global time in distributed systems. In *ICDCS* (1987), pp. 299–306.

[35] Dyer, M. E. Linear time algorithms for two- and three-variable linear programs. *SIAM J. Comput. 13*, 1 (1984), 31–45.

[36] Elson, J., Girod, L., and Estrin, D. Fine-grained network time synchronization using reference broadcasts. In *OSDI '02: Proceedings of the 5th symposium on Operating systems design and implementation* (New York, NY, USA, 2002), ACM, pp. 147–163.

[37] Elteto, T., and Molnar, S. On the distribution of round-trip delays in tcp/ip networks. *Local Computer Networks, Annual IEEE Conference on* (1999), 172.

[38] Fischler, M. A., and Bolles, R. C. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM 24*, 6 (1981), 381–395.

[39] Fitzgibbon, A. W. Robust registration of 2D and 3D point sets. In *Proc. British Machine Vision Conf.* (2001), pp. 662–670.

[40] Friedman, J. H., Bentley, J. L., and Finkel, R. A. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software 3*, 3 (Sept. 1977), 209–226.

[41] Früh, C., and Zakhor, A. Constructing 3D city models by merging aerial and ground views. *IEEE Comput. Graph. Appl. 23*, 6 (2003), 52–61.

[42] Ghosh, A. A FORTRAN program for fitting Weibull distribution and generating samples. *Computers & Geosciences 25*, 7 (1999), 729 – 738.

[43] Gusella, R., and Zatti, S. The Accuracy of the Clock Synchronization Achieved by TEMPO in Berkeley UNIX 4.3BSD. *IEEE Trans. Software Eng. 15*, 7 (1989), 847–853.

[44] Hähnel, D., and Burgard, W. Probabilistic matching for 3D scan registration. In *Fachtagung ROBOTIK* (2002).

[45] Hähnel, D., Burgard, W., and Thrun, S. Learning compact 3D models of indoor and outdoor environments with a mobile robot. *Robotics and Autonomous Systems 44*, 1 (2003), 15–27.

[46] Harrison, A., and Newman, P. High quality 3D laser ranging under general vehicle motion. In *Proc. IEEE International Conference on Robotics and Automation (ICRA'08)* (Pasadena, California, April 2008).

[47] Hartley, R. I., and Zisserman, A. *Multiple View Geometry in Computer Vision*, second ed. Cambridge University Press, ISBN: 0521540518, 2004.

[48] Hernández, J., and Phillips, I. Weibull mixture model to characterise end-to-end internet delay at coarse time-scales. *IEE Proceedings - Communications 153*, 2 (2006), 295–304.

[49] Hoiem, D., Efros, A. A., and Hebert, M. Putting objects in perspective. In *In CVPR* (2006), pp. 2137–2144.

[50] Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., and Stuetzle, W. Surface reconstruction from unorganized points. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1992), ACM Press, pp. 71–78.

[51] Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., and Stuetzle, W. Mesh optimization. *Computer Graphics 27*, Annual Conference Series (1993), 19–26.

[52] Howard, A., Wolf, D., and Sukhatme, G. Towards 3D mapping in large urban environments. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems* (Sendai, Japan, Sept 2004), pp. 419–424.

[53] Huber, D., and Vandapel, N. Automatic 3D underground mine mapping. In *International Conference on Field and Service Robotics* (July 2003).

[54] Huber, P. *Robust Statistics*. Wiley, New York, 1974.

[55] Khlifi, H., and Grégoire, J.-C. Low-complexity offline and online clock skew estimation and removal. *Computer Networks 50*, 11 (2006), 1872–1884.

[56] Knuth, D. E. *The Art of Computer Programming, Volume II: Seminumerical Algorithms, 2nd Edition*. Addison-Wesley, 1981.

[57] Kobbelt, L. P., Botsch, M., Schwanecke, U., and Seidel, H.-P. Feature sensitive surface extraction from volume data. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), ACM Press, pp. 57–66.

[58] Kolluri, R., Shewchuk, J. R., and O'Brien, J. F. Spectral surface reconstruction from noisy point clouds. In *Symposium on Geometry Processing* (July 2004), ACM Press, pp. 11–21.

[59] Lamon, P., Stachniss, C., Triebel, R., Pfaff, P., Plagemann, C., Grisetti, G., Kolski, S., Burgard, W., and Siegwart, R. Mapping with an autonomous car. In *In Proc. of the Workshop on Safe Navigation in Open and Dynamic Environments at the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)* (Beijing, China, 2006).

[60] Lamport, L. Time, clocks, and the ordering of events in a distributed system. *Commun. ACM 21*, 7 (1978), 558–565.

[61] Lamport, L., and Melliar-Smith, P. M. Synchronizing clocks in the presence of faults. *J. ACM 32*, 1 (1985), 52–78.

[62] Lorensen, W. E., and Cline, H. E. Marching cubes: A high resolution 3D surface construction algorithm. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1987), ACM Press, pp. 163–169.

[63] Lu, F., and Milios, E. Globally consistent range scan alignment for environment mapping. *Autonomous Robots 4*, 4 (1997), 333–349.

[64] Lundelius, J., and Lynch, N. A. A new fault-tolerant algorithm for clock synchronization. In *PODC* (1984), pp. 75–88.

[65] Lundelius, J., and Lynch, N. A. An upper and lower bound for clock synchronization. *Information and Control 62*, 2/3 (1984), 190–204.

[66] Martin, M. C., and Moravec, H. Robot evidence grids. Tech. Rep. CMU-RI-TR-96-06, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, March 1996.

[67] Marzullo, K., and Owicki, S. S. Maintaining the time in a distributed system. *Operating Systems Review 19*, 3 (1985), 44–54.

[68] Megiddo, N. Linear-time algorithms for linear programming in r^3 and related problems. *SIAM J. Comput. 12*, 4 (1983), 759–776.

[69] Miller, J. V., Breen, D. E., Lorensen, W. E., O'Bara, R. M., and Wozny, M. J. Geometrically deformed models: a method for extracting closed geometric models form volume data. In *SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1991), ACM Press, pp. 217–226.

[70] Mills, D. Internet time synchronization: The network time protocol. *Communications, IEEE Transactions on 39*, 10 (Oct 1991), 1482–1493.

[71] Montemerlo, M., and Thrun, S. A multi-resolution pyramid for outdoor robot terrain perception. In *Proceedings of the AAAI National Conference on Artificial Intelligence* (San Jose, CA, 2004), AAAI.

[72] Moon, S. B. *Measurement and analysis of end-to-end delay and loss in the internet*. PhD thesis, University of Massachusetts Amherst, 2000. Director-Kurose, James F. and Director-Towsley, Donald F.

[73] Moon, S. B., Skelly, P., and Towsley, D. F. Estimation and removal of clock skew from network delay measurements. In *INFOCOM* (1999), pp. 227–234.

[74] Mount, D. M. *ANN Programming Manual*, 2005.

[75] Nelder, J. A., and Mead, R. A simplex method for function minimization. *The Computer Journal 7*, 4 (January 1965), 308–313.

[76] Newman, P., Sibley, G., Smith, M., Cummins, M., Harrison, A., Mei, C., Posner, I., Shade, R., Schröter, D., Murphy, L., Churchill, W., Cole, D., and Reid, I. Navigating, recognising and describing urban spaces with vision and laser. *The International Journal of Robotics Research 28* (October 2009).

[77] Newman, P. M. Moos - a mission oriented operating suite. Tech. Rep. OE2003-07, MIT Department of Ocean Engineering, Cambridge, MA, 2003.

[78] Ng, T. C., Guzman, J. I., and Tan, J. C. Development of a 3D ladar system for autonomous navigation. In *Proc. IEEE Robotics, Automation and Mechatronics Conference* (2004).

[79] Nguyen, V., Martinelli, A., Tomatis, N., and Siegwart, R. A comparison of line extraction algorithms using 2D laser rangefinder for indoor mobile robotics. In *Proc. Intelligent Robots and Systems* (2005).

[80] Nielson, G. M., and Hamann, B. The asymptotic decider: Resolving the ambiguity in marching cubes. In *IEEE Visualization* (1991), pp. 83–93.

[81] Ning, P., and Bloomenthal., J. An evaluation of implicit surface tilers. *IEEE Computer Graphics and Applications 13*, 6 (1993), 33–41.

[82] Noh, K.-L., Chaudhari, Q., Serpedin, E., and Suter, B. Novel clock phase offset and skew estimation using two-way timing message exchanges for wireless sensor networks. *Communications, IEEE Transactions on 55*, 4 (April 2007), 766–777.

[83] Nüchter, A., Lingemann, K., Hertzberg, J., and Surmann, H. 6D slam - 3D mapping outdoor environments. *J. Field Robotics 24*, 8-9 (2007), 699–722.

[84] Nüchter, A., Surmann, H., Lingemann, K., and Hertzberg, J. Semantic scene analysis of scanned 3D indoor environments. In *VMV 2003: Proceedings of the Vision, Modeling, and Visualization conference* (2003).

[85] Ohtake, Y., Belyaev, A., Alexa, M., Turk, G., and Seidel, H.-P. Multi-level partition of unity implicits. *ACM Trans. Graph. 22*, 3 (2003), 463–470.

[86] PalChaudhuri, S., Saha, A. K., and Johnson, D. B. Adaptive clock synchronization in sensor networks. In *IPSN '04: Proceedings of the 3rd international symposium on Information processing in sensor networks* (New York, NY, USA, 2004), ACM, pp. 340–348.

[87] Papagiannaki, K., Moon, S., Fraleigh, C., Thiran, P., and Diot, C. Measurement and Analysis of Single-Hop Delay on an IP Backbone Network. *IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS 21*, 6 (2003), 908–921.

[88] Pauly, M., Gross, M., and Kobbelt, L. P. Efficient simplification of point-sampled surfaces. In *VIS '02: Proceedings of the conference on Visualization '02* (Washington, DC, USA, 2002), IEEE Computer Society, pp. 163–170.

[89] Paxson, V. On calibrating measurements of packet transit times. In *SIGMETRICS* (1998), pp. 11–21.

[90] Posner, I., Schroeter, D., and Newman, P. Online generation of scene descriptions in urban environments. *Robotics and Autonomous Systems 56*, 11 (2008), 901 – 914. Semantic Knowledge in Robotics.

[91] Press, W., Teukolsky, S., Vetterling, W., and Flannery, B. *Numerical Recipes in C*, 2nd ed. Cambridge University Press, Cambridge, UK, 1992.

[92] Pulli, K., Duchamp, T., Hoppe, H., McDonald, J., Shapiro, L., and Stuetzle, W. Robust meshes from multiple range maps. In *NRC '97: Proceedings of the International Conference on Recent Advances in 3-D Digital Imaging and Modeling* (Washington, DC, USA, 1997), IEEE Computer Society, p. 205.

[93] Rusinkiewicz, S., and Levoy, M. Efficient variants of the ICP algorithm. In *Third International Conference on 3D Digital Imaging and Modeling (3DIM)* (June 2001).

[94] Saxena, A., Chung, S. H., and Ng, A. Y. 3-D depth reconstruction from a single still image. *Int. J. Comput. Vision 76*, 1 (2008), 53–69.

[95] Scaramuzza, D., Harati, A., and Siegwart, R. Extrinsic self calibration of a camera and a 3D laser range finder from natural scenes. In *IEEE International Conference on Intelligent Robots and Systems* (2007).

[96] Schaefer, S., and Warren, J. D. Dual marching cubes: Primal contouring of dual grids. In *Pacific Conference on Computer Graphics and Applications* (2004), pp. 70–76.

[97] Scharstein, D., and Szeliski, R. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Int. J. Comput. Vision 47*, 1-3 (2002), 7–42.

[98] Sequeira, V., Ng, K., Wolfart, E., Goncalves, J. G. M., and Hogg, D. Automated 3D reconstruction of interiors with multiple scan-views. In *Proc. of SPIE, Electronic Imaging '99, The Society for Imaging Science and Technology /SPIE's 11th Annual Symposium.* (San Jose, CA, USA, January 1999, 1999).

[99] Sequeira, V., Ng, K., Wolfart, E., Goncalves, J. G. M., and Hogg, D. Automated reconstruction of 3D models from real environments. *ISPRS Journal of Photogrammetry and Remote Sensing 54*, 1 (1999), 1–22.

[100] Shekhar, R., Fayyad, E., Yagel, R., and Cornhill, J. F. Octree-based decimation of marching cubes surfaces. In *VIS '96: Proceedings of the 7th conference on Visualization '96* (1996), IEEE Computer Society Press, pp. 335–ff.

[101] Sibson, R. A brief description of natural neighbour interpolation. In *Interpolating Multivariate Data*, V. Barnet, Ed. John Wily & Sons, Chichester, 1981, pp. 21–36.

[102] SICK. *Telegrams for Operating/Configuring the LMS 2xx Laser Measurement Systems.* SICK AG, Germany, Apr. 2003.

[103] Sirdey, R., and Maurice, F. A linear programming approach to highly precise clock synchronization over a packet network. *4OR 6*, 4 (2008), 393–401.

[104] Smith, M., Baldwin, I., Churchill, W., Paul, R., and Newman, P. The new college vision and laser data set. *The International Journal of Robotics Research 28*, 5 (May 2009), 595 – 599. Data Papers - Peer Reviewed Publication of High Quality Data Sets.

[105] Surmann, H., Lingemann, K., Nüchter, A., and Hertzberg, J. A 3D laser range finder for autonomous mobile robots. In *Proc. 32nd International Symposium on Robotics (ISR'01)* (2001).

[106] Thrun, S., Fox, D., and Burgard, W. A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping. In *IEEE Int. Conf. on Robotics and Automation* (2000).

[107] Tomasi, C., and Manduchi, R. Bilateral filtering for gray and color images. In *Computer Vision, 1998. Sixth International Conference on* (Jan 1998), pp. 839–846.

[108] Torres-Méndez, L. A., and Dudek, G. Statistics of visual and partial depth data for mobile robot environment modeling. In *MICAI* (2006), pp. 715–725.

[109] Triebel, R., and Burgard, W. Using hierarchical EM to extract planes from 3D range scans. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)* (2005).

[110] Tsai, R. Y. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE Journal of Robotics and Automation* (1987), 323–344.

[111] Tuley, J., Vandapel, N., and Hebert, M. Analysis and removal of artifacts in 3-D ladar data. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)* (2005), pp. 2215–2222.

[112] Underwood, J. P., Hill, A. J., and Scheding, S. Calibration of range sensor pose on mobile platforms. In *IROS* (2007), IEEE, pp. 3866–3871.

[113] Unnikrishnan, R., and Hebert, M. Fast extrinsic calibration of a laser rangefinder to a camera. Tech. Rep. CMU-RI-TR-05-09, Robotics Institute, Pittsburgh, PA, July 2005.

[114] Veitch, D., Ridoux, J., and Korada, S. B. Robust synchronization of absolute and difference clocks over networks. *IEEE/ACM Trans. Netw. 17*, 2 (2009), 417–430.

[115] Wang, J., Zhou, M., and Zhou, H. Clock synchronization for internet measurements: a clustering algorithm. *Computer Networks 45*, 6 (2004), 731–741.

[116] Weibull, W. A statistical distribution function of wide applicability. *Journal of Applied Mechanics, Transactions ASME 18*, 3 (September 1951), 293–297.

[117] Weingarten, J. *Feature-based 3D SLAM.* PhD thesis, EPFL, Lausanne, 2006.

[118] Wildes, D. G. Digital computer sliding-window minimum filter. In *US Patent 5319583* (June 1994).

[119] Wong, U., Garney, B., Whittaker, W., and Whittaker, R. Camera and lidar fusion for mapping of actively illuminated subterranean voids. In *Proceedings of the 7th International Conference on Field and Service Robotics* (Cambridge, Massachusetts, July 2009).

[120] Woodford, O. J., Torr, P. H. S., Reid, I. D., and Fitzgibbon, A. W. Global stereo reconstruction under second order smoothness priors. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR 2008)* (Anchorage, Alaska, June 24-26 2008).

[121] Wulf, O., and Wagner, B. Fast 3D scanning methods for laser measurement systems. In *Int. Conf. on Control Systems and Computer Sci* (Bucharest, Romania, July 2003), vol. 1, pp. 312–317.

[122] Yang, Q., Yang, R., Davis, J., and Nister, D. Spatial-depth super resolution for range images. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on.* (2007), 1–8.

[123] Zhang, L., Liu, Z., and Honghui Xia, C. Clock synchronization algorithms for network measurements. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies.* (2002), vol. 1, IEEE, pp. 160–169 vol.1.

[124] Zhang, Z. A flexible new technique for camera calibration. *IEEE Trans. Pattern Anal. Mach. Intell. 22*, 11 (2000), 1330–1334.

[125] Zhao, H., Chen, Y., and Shibasaki, R. An efficient extrinsic calibration of a multiple laser scanners and cameras' sensor system on a mobile platform. In *Intelligent Vehicles Symposium, 2007 IEEE* (June 2007), pp. 422–427.